高职高专计算机专业精品教材

中小型Web项目开发实战

冯艳玲　张晖　邓果丽　易海涛　编　著



清华大学出版社

高职高专计算机专业精品教材

中小型 Web 项目开发实战

冯艳玲　张　晖　邓果丽　易海涛　主编

清华大学出版社

北　京

内 容 简 介

本书采用了“项目引导，任务驱动”的组织结构，通过建立一个“连锁店进、销、存管理系统”项目，系统地介绍了项目开发从需求分析到编程实现的完整流程，整个项目贯穿讲解并应用了开发基于JSP＋JavaBenas＋Servlet的Web中小型项目所需要的知识点和技能。

本书分为四个部分。第一部分为项目的准备阶段；第二部分为系统数据访问功能模块的设计开发；第三部分为系统的安全设计；第四部分为数据分析。

本书共设置了难易不同的30多个“任务”，这些任务之间有着渐进的关系，建议教师在设备条件许可的情况下，采用“讲练结合”的方式进行授课，课程学时设置每周至少为4学时。

本书可以作为高职高专院校计算机相关专业的教材，也可以作为Web开发工程师的参考资料。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目(CIP)数据

中小型Web项目开发实战/冯艳玲等编著. —北京：清华大学出版社，2013

高职高专计算机专业精品教材

ISBN 978-7-302-32836-0

Ⅰ. ①中…　Ⅱ. ①冯…　②…　Ⅲ. ①主页制作－程序设计－高等职业教育－教材　Ⅳ. ①TP393.092

中国版本图书馆CIP数据核字(2013)第136345号

责任编辑：张龙卿
封面设计：徐日强
责任校对：李　梅
责任印制：李红英

出版发行：清华大学出版社
　　网　　址：http://www.tup.com.cn，http://www.wqbook.com
　　地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
　　社 总 机：010-62770175　　**邮　　购**：010-62786544
　　投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
　　质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
　　课 件 下 载：http://www.tup.com.cn，010-62795764
印 装 者：北京国马印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：13.5　　**字　　数**：299千字
版　　次：2013年9月第1版　　**印　　次**：2013年9月第1次印刷
印　　数：1～3000
定　　价：28.00元

产品编号：053749-01

前　言

本书详细介绍了中小型 Java Web 项目开发流程中所需掌握的基础知识和基本技能，主要内容包括项目需求分析的撰写、搭建 Java Web 项目开发环境、MySQL 数据库的设计与实现、使用连接池技术连接数据库、增删改查数据、系统的安全性设计、报表系统的设计与实现。全书以“佳衣屋连锁店进、销、存管理系统”作为案例，完整地再现了基于 mvc 设计模式的 Java Web 中小型项目开发流程，重点培养读者掌握项目开发过程中所需的技能和应遵守的业内规范。

本书采用了“项目引领，任务驱动”的教学模式，按照中小型 Java Web 项目从设计到开发、实现的一般性流程组织本书的内容。书中技能的讲授避免了艰涩的理论说教，每个“任务”都有大量的 step by step 实操图示向导指引和源代码示例，按照向导的流程进行说明。本书采用了“归纳法”的知识学习方法，每章的最后一节为本章的“课后练习”，在完成前序小节各个“任务”的基础上，读者对知识点已经有了感性认识，通过本练习，读者对一些理论性的知识可更快地领悟和掌握。

本书内容实用，可操作性强，可作为计算机软件、计算机信息管理、计算机网络及其他计算机相关专业的高职高专教材或者实训指导书，同时也可作为 Java 编程爱好者及初中级 Java Web 开发人员的参考用书。

由于编者水平有限，加之时间仓促，书中难免有不成熟之处，欢迎读者批评指正，以便在今后的修订中加以完善。

编　者

2013 年 5 月

目　录

第一部分　项目的准备阶段

第二部分 系统数据访问功能模块的设计开发



第一部分

项目的准备阶段



第 1 章　项目需求分析与设计

本章学习要点：

- 了解当前主流 Web 项目开发技术及其特点；
- 掌握与客户沟通的能力；
- 熟练掌握项目需求分析报告的编写规范。

1.1　开发技术的选取

近年来，我国软件产业发展始终保持较快的增长。基于 Web 的应用软件由于具有投入成本比较小、软件开发比较容易、对企业快速扩张支持较好等特点，成为软件开发领域的发展趋势。目前主流的 Web 开发技术有如下几种。

（1）PHP。PHP 是英文超级文本预处理语言 Hypertext Preprocessor 的缩写。PHP 是一种 HTML 内嵌式的语言，是一种在服务器端执行的嵌入 HTML 文档的脚本语言，语言的风格类似于 C 语言，被广泛地运用。

（2）dotNET。dotNET 就是.NET，严格地说是.Net Framework 框架。.NET 是一个微软开发的编程环境，可以使用 ASP、C#（# 读作 sharp）、VB 等多种编程语言。

.NET 是 Microsoft 用于创建 XML Web 服务（下一代软件）的平台，借助于该平台，可以创建和使用基于 XML 的应用程序、进程和 Web 站点以及服务。借助于微软强大的推广能力和快速的版本更新，.NET 已经迅速成为主流的 Web 开发技术。

（3）Java Web 技术。以 Java 为基础的 Web 开发技术，统称为 Java Web 技术。由于 Java 语言本身的安全性设计，使得 Java Web 的安全性较好。本书中的中小型 Web 项目的开发就是采用了 Java Web 技术。

这三种开发技术没有严格的“优”与“劣”之分，Web 项目的开发到底采用何种技术，是项目的规模、安全性要求、预算甚至开发人员的使用习惯等因素权衡之下的考量。

TIOBE 编程语言社区排行榜是编程语言流行趋势的一个指标，每月更新。排行榜排名基于互联网上有经验的程序员、课程和第三方厂商的数量。排名使用著名的搜索引擎（诸如 Google、MSN、雅虎）以及 Wikipedia 和 YouTube 进行计算。

表 1-1 是 2013 年 5 月编程语言的排行，可以看到三种主流的 Web 开发技术在语言使用排行榜上均排名靠前。

表 1-1　TIOBE 编程语言排行

Position May 2013	Position May 2012	Delta in Position	Programming Language	Ratings May 2013	Delta May 2012	Status
1	1	=	C	18.729%	+1.38%	A
2	2	=	Java	16.914%	+0.31%	A
3	4	⬆	Objective-C	10.428%	+2.12%	A
4	3	⬇	C++	9.198%	−0.63%	A
5	5	=	C#	6.119%	−0.70%	A
6	6	=	PHP	5.784%	+0.07%	A

读者可以通过 http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html 网址查看最新的 TIOBE 排名。

1.2　Java Web 开发模式

根据所开发的项目的规模，可采用以下不同的 Java Web 开发模式。

(1) 直接使用 JSP。对于小型 Web 站点，可以直接使用 JSP 来构建动态网站。

(2) JSP+JavaBeans。在该模式中，JSP 页面独自响应请求并将处理结果返回客户。所有的数据通过 Bean 来处理，JSP 实现页面的表现。该模式也实现了页面的表现和页面商业逻辑的分离，可以很好地满足小型应用的需要。

(3) JSP+JavaBeans+Servlet。这种模式的主要思想是使用一个或多个 Servlet 作为控制器。多个 Servlet 控制器可以结合起来完成复杂的任务，可重用性好，可用作中小型项目的开发模式。

(4) Java EE 开发模式。Java EE 基于标准的体系结构和组件开发，采用了松散的设计方法，组件既可以单独调用，也可以组合调用。Java EE 组件解决了所有底层复杂的问题，组件易于升级。组件和门户基于 XML 配置的方式，可以灵活配置。能提供良好的可开发系统外部接口，适合开发中到大型网站项目。

本书将带领读者一同为“佳衣屋”公司开发一个基于 JSP+JavaBeans+Servlet 技术的中型信息系统，在软件项目进入项目开发之前要进行一系列的准备工作，包括与用户充分沟通并进行需求分析、搭建开发和测试环境以及设计数据库等工作。

1.3　软件项目的需求分析

按照软件产品开发过程的技术特点，可将软件分为定制软件(customized software)和通用软件(packaged software)。其中，定制软件是按照单个客户的个性化要求，以软件项目的方式为其提交个性化的解决方案。通用软件是以通用软件包的方式提交给不同的

用户来使用。Web项目多为定制软件,需要根据用户的具体情况、具体要求进行设计,并提供相应的服务。

需求分析是定制Web项目开发的第一步,也是非常关键的一步。需求分析是指开发人员与用户进行沟通,理解用户需求,就软件功能与客户达成一致,最终形成开发计划的一个过程。需求分析的结果可以形成《需求分析报告》。用户确认并签署《需求分析报告》后,项目就可进入开发阶段了。

1.4 需求分析报告的格式

需求分析报告因所开发的软件项目类型的不同会有所区别,本节介绍中小型Web项目的需求分析报告的常用格式。

1.4.1 编写目的

需求分析报告的引言部分一般是阐述项目开发的目的和意义,使阅读者对项目有一个整体的了解。

示例1-1 “佳衣屋”项目需求分析报告中的编写目的

“佳衣屋”信息系统需求分析报告

1 编写目的

本报告的目的是根据客户的需求,对系统功能、性能需求向“佳衣屋”公司、项目组开发成员、项目实施组和测试成员提供一个清晰的陈述,为项目的后续展开提供指导。

本报告主要阐述“佳衣屋”公司服装连锁销售的进、销、存业务流程,兼顾员工的管理和报表分析等功能。项目上线运行后,信息系统的使用者可以及时了解商品的销售、库存等信息,并能够及时地对商品进行调度,有助于公司的运营和管理。

1.4.2 运行环境

这一部分说明项目的开发和运行所需的软、硬件条件。

示例1-2 “佳衣屋”项目需求分析报告中的运行环境

2 运行环境

硬件:机架式服务器

操作系统:Windows Server 2003

支撑环境:JDK1.6,MySQL 5.1,Tomcat 7.0

其他相关软件:MyEclipse 9.0,SQLyog 9.0

1.4.3 系统结构分析

一个软件系统是由许多功能模块构成的。模块化是一种重要的设计思想，这种思想把一个复杂的系统分解为一些规模较小、功能较简单、更易于建立和修改的部分，各个模块具有相对独立性，可以分别加以设计实现。

系统结构通常以树形框图即功能结构图的形式来描述模块之间的关系和相互作用，图中的每一个框代表一个功能模块。每个功能模块还可以继续分解为第三层、第四层甚至更多的子功能模块。

示例 1-3 “佳衣屋”项目的系统功能结构图如图 1-1 所示

3 系统结构分析

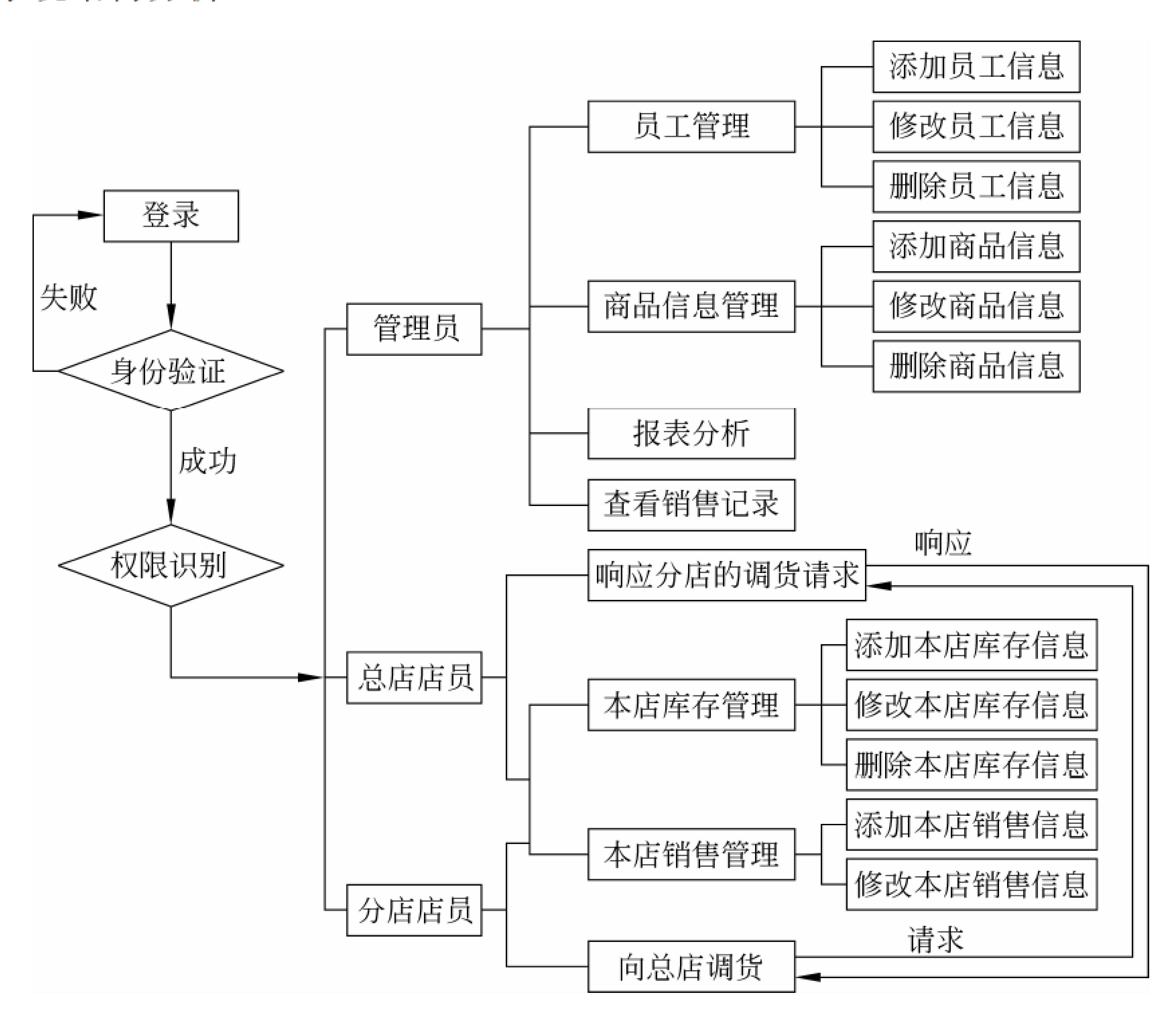


图 1-1 “佳衣屋”项目需求分析中的功能结构图

1.4.4 系统功能分析

系统功能分析部分是把功能模块做一个比较详细的说明，用来指导编程实现。

示例 1-4 “佳衣屋”项目需求分析报告中的系统功能分析部分

4 系统功能分析

4.1 登录模块

用户提交用户名、密码、验证码、选择权限（管理员/总店店员/分店店员）、选择是否记

住密码，单击“登录”按钮进行身份验证，单击“重置”按钮则清空用户输入的信息。用户界面如图 1-2 所示。

网站Logo

用户名：
密　码：
验证码：
权　限：　(列表菜单)
是否记住密码：　(列表菜单)

登录　重置

图 1-2　“佳衣屋”项目登录页示意图

（略）

4.2　管理员权限下的功能模块

4.2.1　商品管理模块

1．添加商品子模块

管理员单击左侧功能导航栏中的“商品管理”下的“添加商品信息”超链接，在右侧栏中出现填写商品信息的表单，输入商品的款号、名称、颜色、销售指导价并上传商品图片，单击“添加”按钮后，将新商品的基本信息添加到系统中。用户界面如图 1-3 所示。

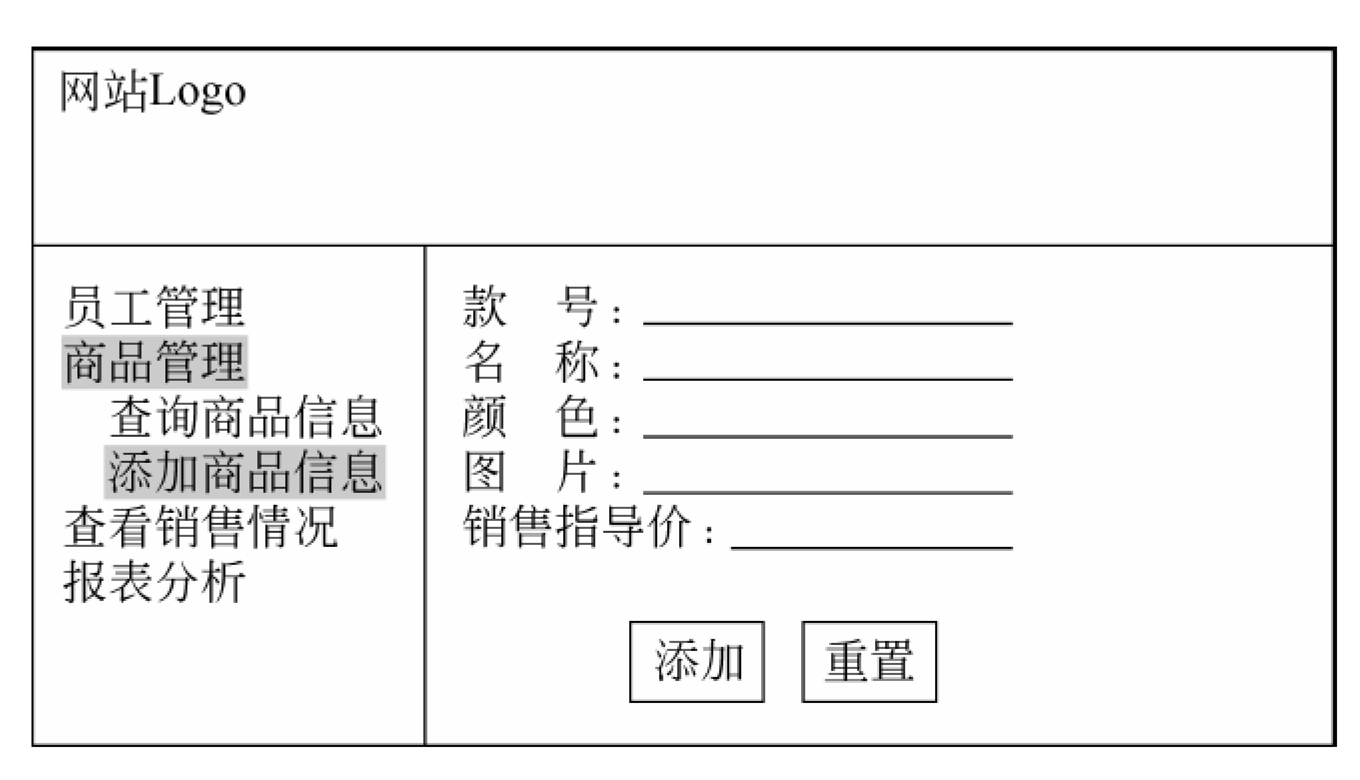


图 1-3　“佳衣屋”项目管理员添加商品页示意图

（略）

4.3　分店店员权限下的功能模块

（略）

4.3.3　进货模块

分店店员从总店库存商品列表中选中需要调入的商品，并填写每种商品的进货量，单击“进货”按钮后完成批量进货请求的发送。用户界面如图 1-4 所示。

网站Logo

库存管理
销售管理
进　　货
进货查询

款号	名称	类型	码数	颜色	售价	库存量	进货量
□ xxx	xxx	xx	xx	xx	xx	xxxxx	______
□ yyy	yyy	yy	yy	yy	yy	yyyyy	______
□ zzz	zzzz	zz	zz	zz	zz	zzzzz	______

进货

图 1-4　分店店员从总店进货页示意图

（略）

4.4　总店店员权限下的功能模块

4.4.1　进货请求处理模块

总店店员单击左侧功能导航栏中的“发货”超链接，在右侧栏中出现各分店的进货请求及当前处理的阶段。

总店店员单击操作栏中的“确认”按钮，表示对进货请求予以确认。如果库存数量大于进货数量，则该条进货请求的操作栏出现“发货”按钮，分店的该条进货请求状态将变为“待发货”；如果库存数量小于进货数量，则提示“库存不足”，总店店员可单击“取消”按钮来取消此次请求，总店和分店的该条进货请求状态将变为“已取消”，或者待库存增加后再确认此进货请求。

总店店员单击操作栏中的“发货”按钮，表示已将商品发往分店，总店和分店的该条进货请求状态将变为“已发货”。用户界面如图 1-5 所示。



图 1-5　分店店员从总店进货页示意图

因为篇幅的限制，本章中的需求分析及本书后面的章节中的示例，仅能包含一个真实连锁销售进、销、存管理系统中的部分功能，希望随着章节的推进，读者能够学以致用，充分发挥主动性，将这一系统完善。

课 后 练 习

1. 读者可以为“佳衣屋”项目添加功能模块，并在系统结构图中画出。

2. 请自由组织起来到企事业单位进行实地考察，了解它们的经营或者管理流程，以及信息化需求，写出需求分析报告。

第 2 章　MyEclipse 集成开发环境

本章学习要点：

- 熟练掌握 MyEclipse 集成开发环境(IDE)中创建 Web 项目的方法；
- 熟练掌握 MyEclipse 集成开发环境中项目的调试方法。

Java Web 项目的开发和运行，需要安装必要的软件，并进行一定的配置。

2.1　任务一： JDK 的安装和配置

2.1.1　JDK 的下载和安装

学习 Java 语言的读者对 JDK(Java Development Kit)一定不陌生，JDK 包含了多个用于 Java 开发的组件，其中包括以下工具。

- javac：编译器，将后缀名为.java 的源代码编译成后缀名为.class 的字节码。
- Java：运行工具，运行.class 的字节码。
- jar：打包工具，将相关的类文件打包成一个文件。
- javadoc：文档生成器，从源码注释中提取文档，注释需符合规范。
- jdb debugger：调试工具。

JDK 中还包括完整的 JRE(Java Runtime Environment，Java 运行环境)，也被称为 private runtime，它包括了用于产品环境的各种库类，如基础类库 rt.jar，以及给开发人员使用的补充库，如国际化与本地化的类库、IDL 库等。

JDK 中还包括各种样例程序，用以展示 Java API 中的各个部分。

可以从 Oracle 的官方网站(http://www.oracle.com/)上搜索并下载最新的适用于 Windows 平台的 JDK 版本，如图 2-1 所示，并按照安装向导的提示进行安装即可。

2.1.2　JDK 环境变量的设置

JDK 安装完成后，还要进行环境变量的设置。步骤如下。

(1) 单击系统的“开始”菜单，打开“控制面板”，双击“系统”工具，单击任务中的“高级系统设置”，如图 2-2 所示。

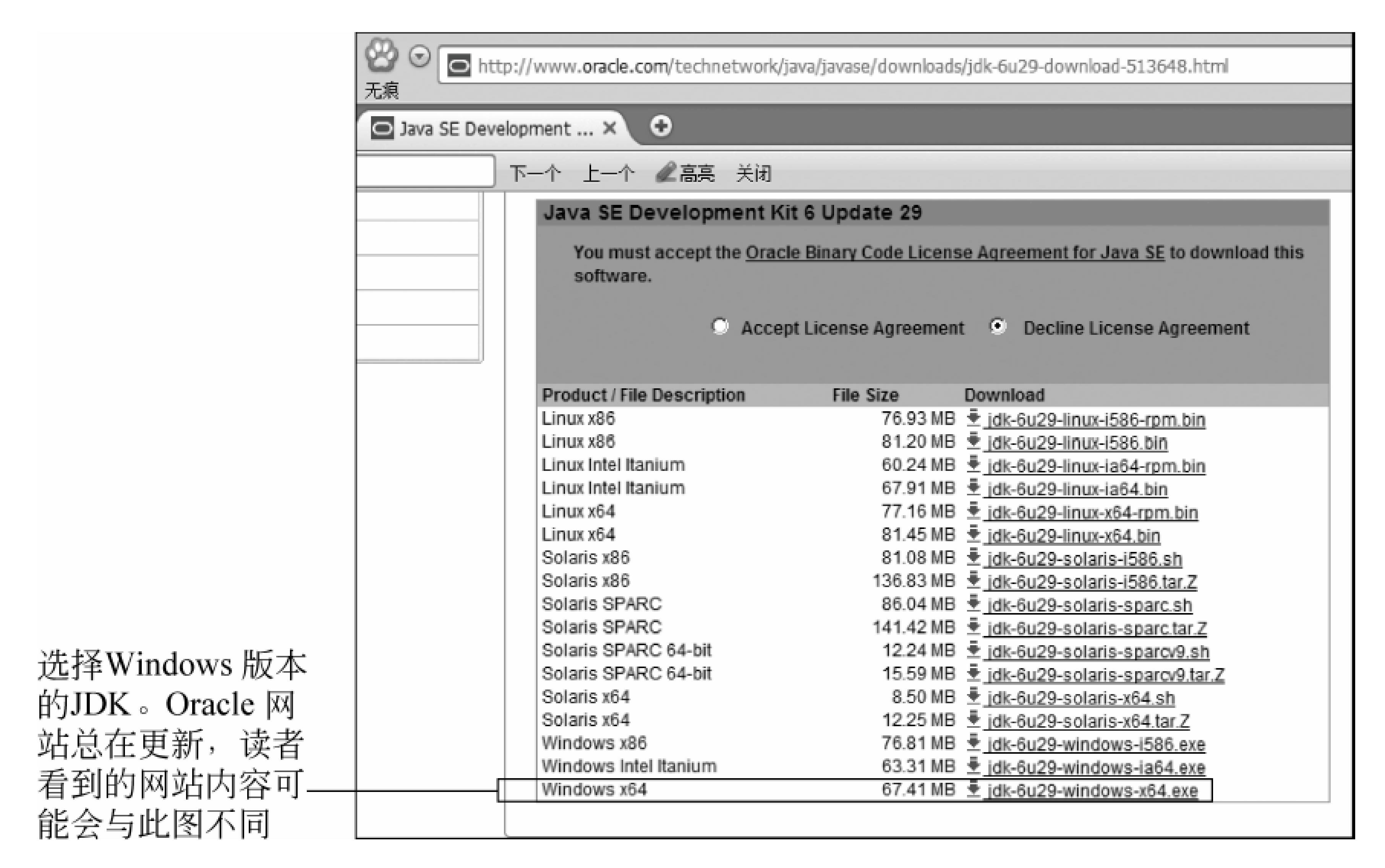


图 2-1　从 Oracle 网站下载 JDK

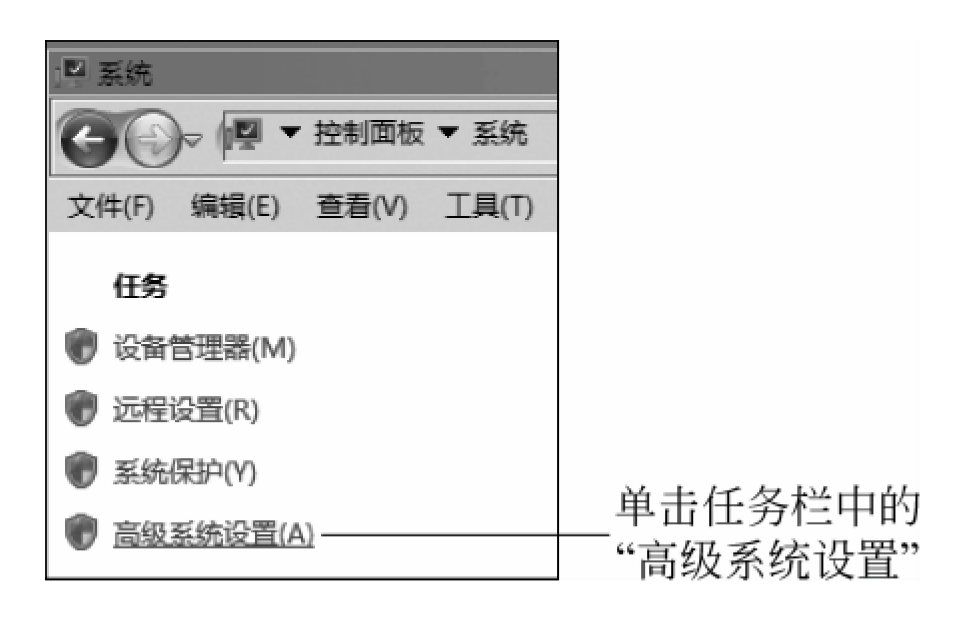


图 2-2　“系统”工具中的任务栏

（2）单击弹出的“系统属性”对话框中的“高级”选项卡，之后单击该选项卡中的“环境变量”按钮，可以对环境变量进行查看，如图 2-3 所示。

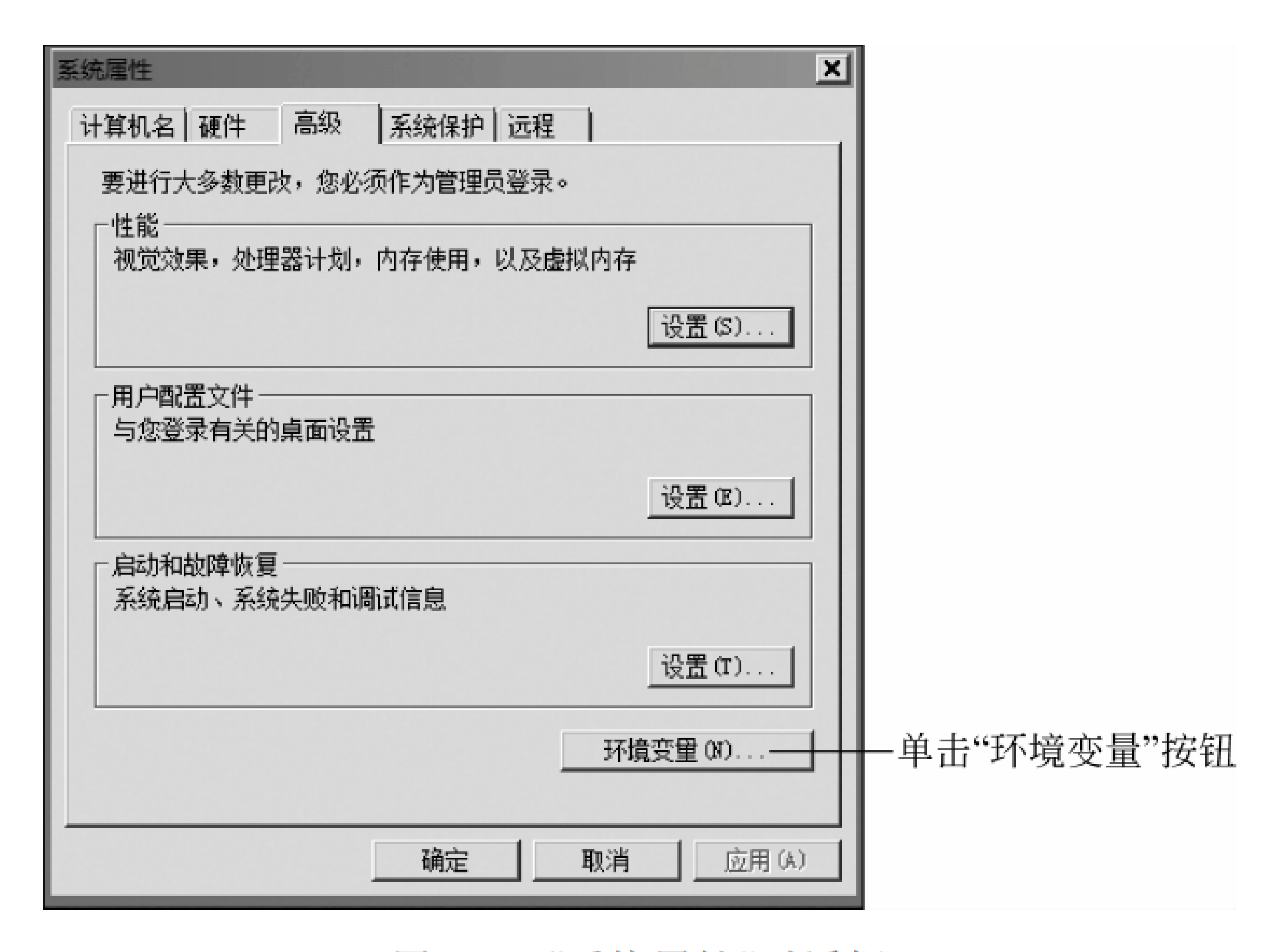


图 2-3　“系统属性”对话框

（3）单击“环境变量”对话框下方的“系统变量”区域中的“新建”按钮，进入环境变量的编辑状态，如图 2-4 所示。

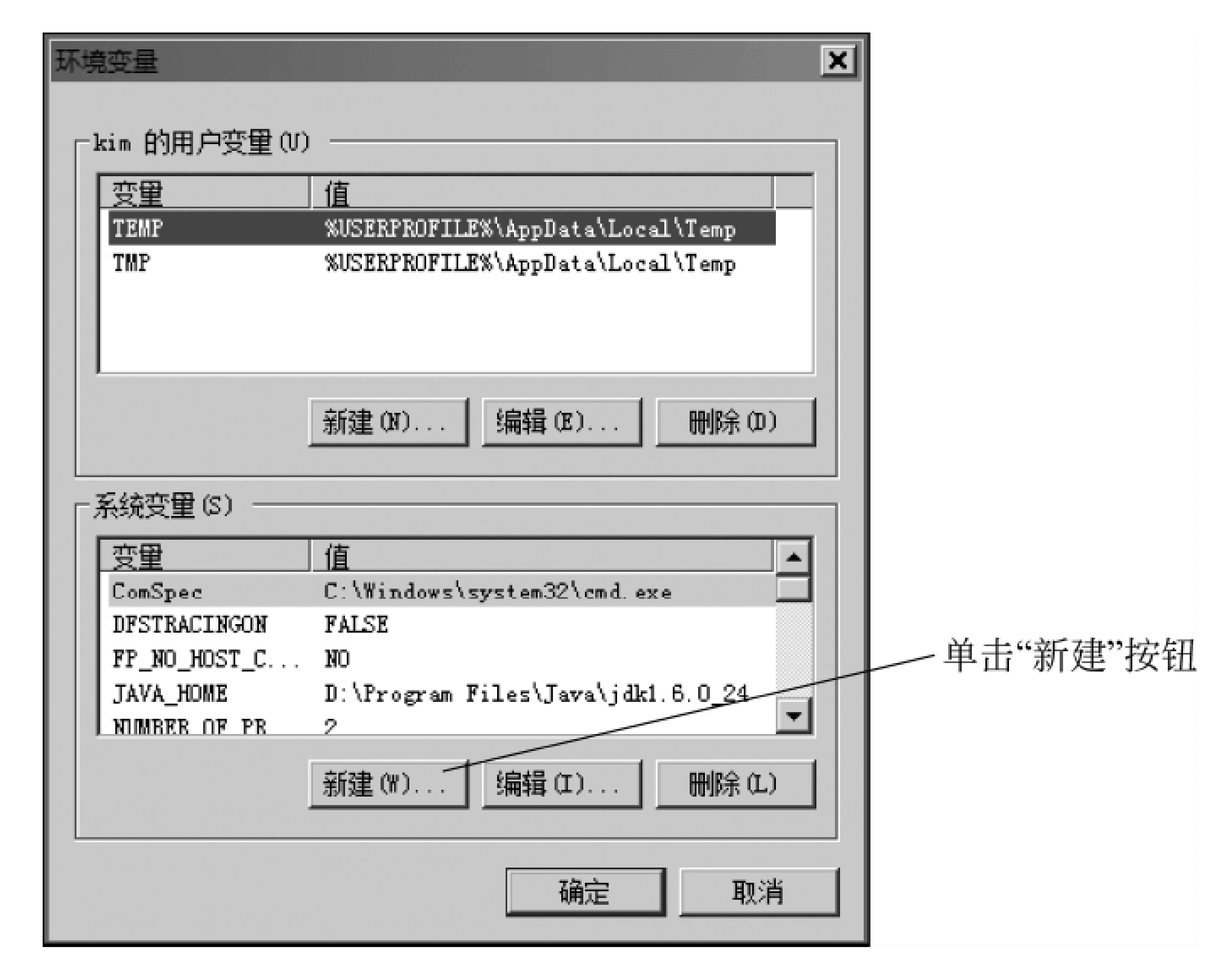


图 2-4　“环境变量”对话框

（4）在“编辑系统变量”对话框中的“变量名”文本框中输入 java_home；“变量值”文本框中输入 JDK 所在的目录，单击“确定”按钮。图 2-5 所示的例子是将 JDK 1.6 安装在 D 盘的\Program Files 目录下。读者应根据实际安装情况来填写变量值，切莫一成不变。

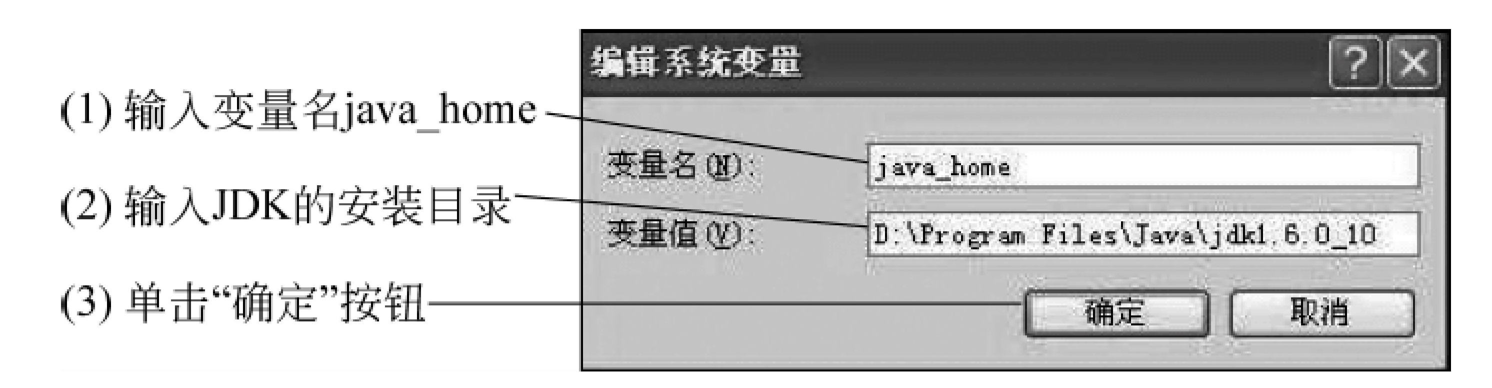


图 2-5　设置系统变量 java_home

（5）检查一下目前系统有无名叫 classpath 的变量，如果有，则选中它并单击“编辑”按钮，如图 2-6 所示。在弹出的“编辑系统变量”对话框中，添加变量值“；%java_home%\lib；%java_home%\lib\tools.jar”，应注意不同的变量间应用分号隔开，之后单击“确定”按钮，如图 2-7 所示；如果系统当前没有 classpath 变量，则需要按照步骤（4）新建一个 classpath 变量。

（6）单击系统变量中的 path 变量，单击“编辑”按钮，如图 2-8 所示。在弹出的“编辑系统变量”对话框中，添加“%java_home%\bin；%java_home%\jre6\bin”，单击“确定”按钮，如图 2-9 所示。

设置完环境变量后，进入命令行模式，输入命令 java -version。如果命令得到响应，系统会输出 JDK 的版本号，如图 2-10 所示，则表明 JDK 的安装和环境的设置已经正确完成了。



图 2-6　编辑系统变量 classpath



图 2-7　将 JDK 的 jar 包添加到系统的 classpath 变量中



图 2-8　编辑系统变量 Path



图 2-9　将 JDK 的可执行文件目录添加到系统的 Path 变量中

```
C:\Windows\system32\cmd.exe
C:\Users\kim>
C:\Users\kim>java -version
java version "1.6.0_24"
Java(TM) SE Runtime Environment (build 1.6.0_24-b07)
Java HotSpot(TM) Client VM (build 19.1-b02, mixed mode, sharing)
```

图 2-10　检验 JDK 的安装与环境变量的设置

2.2　任务二：Tomcat 的安装和配置

Tomcat 是目前比较流行的 Web 应用服务器之一，是一个开源项目，运行时占用的系统资源小，扩展性好，支持负载平衡与邮件服务等开发应用系统常用的功能。

Tomcat 是一个轻量级的应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试 Java Web 项目的首选。

任务目标：

下载并安装 Tomcat 服务器。安装完成后启动服务器，在浏览器地址栏中输入正确的地址，可以看到 Tomcat 默认的主页。

Tomcat 的最新版本可以从 Apache 的网站上下载，如图 2-11 所示。

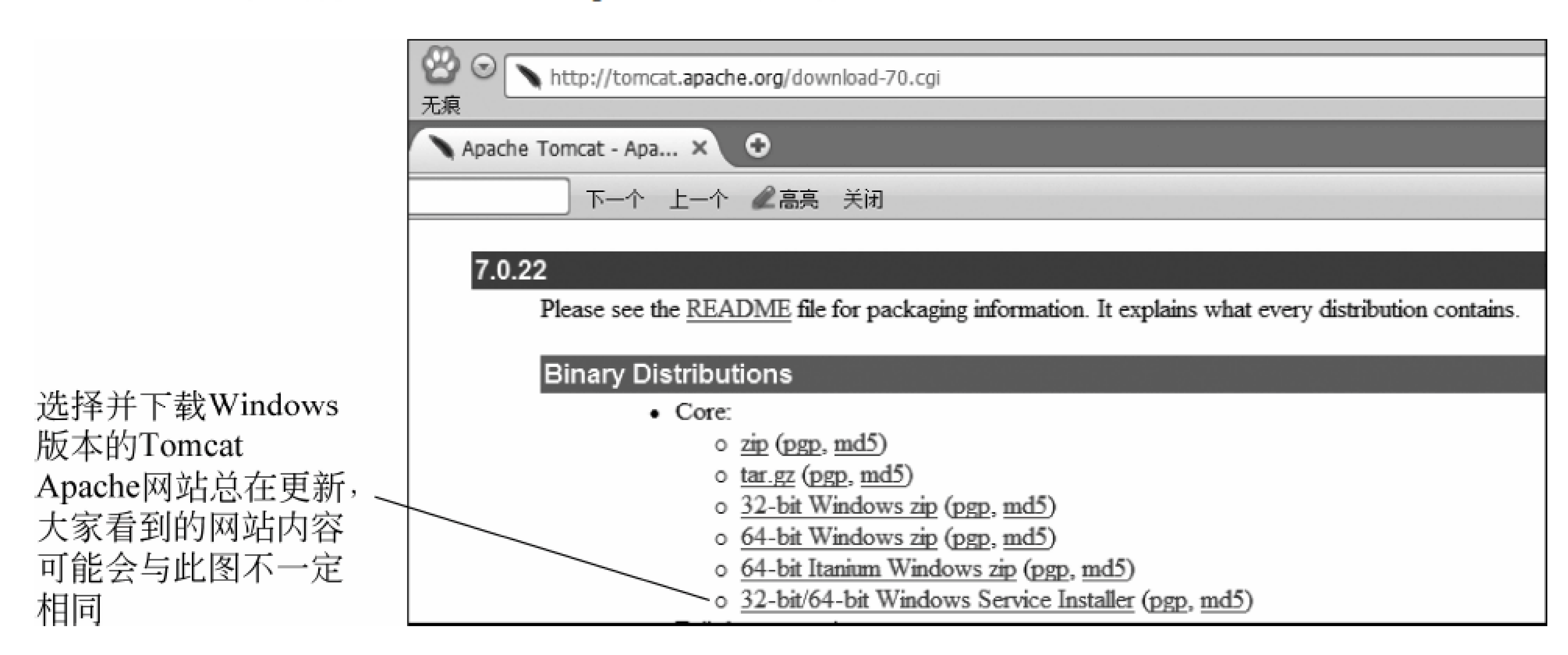


图 2-11　从 Apache 网站下载 Tomcat 7.0

双击下载的安装软件进行安装。Tomcat 的安装很简单，可以按照安装向导的指引完成，需要注意的是配置选项页的设置，如图 2-12 所示。Tomcat 默认的端口号是 8080，当服务器上的 8080 端口已被占用时，这个端口号需要被修改，以免造成端口冲突。

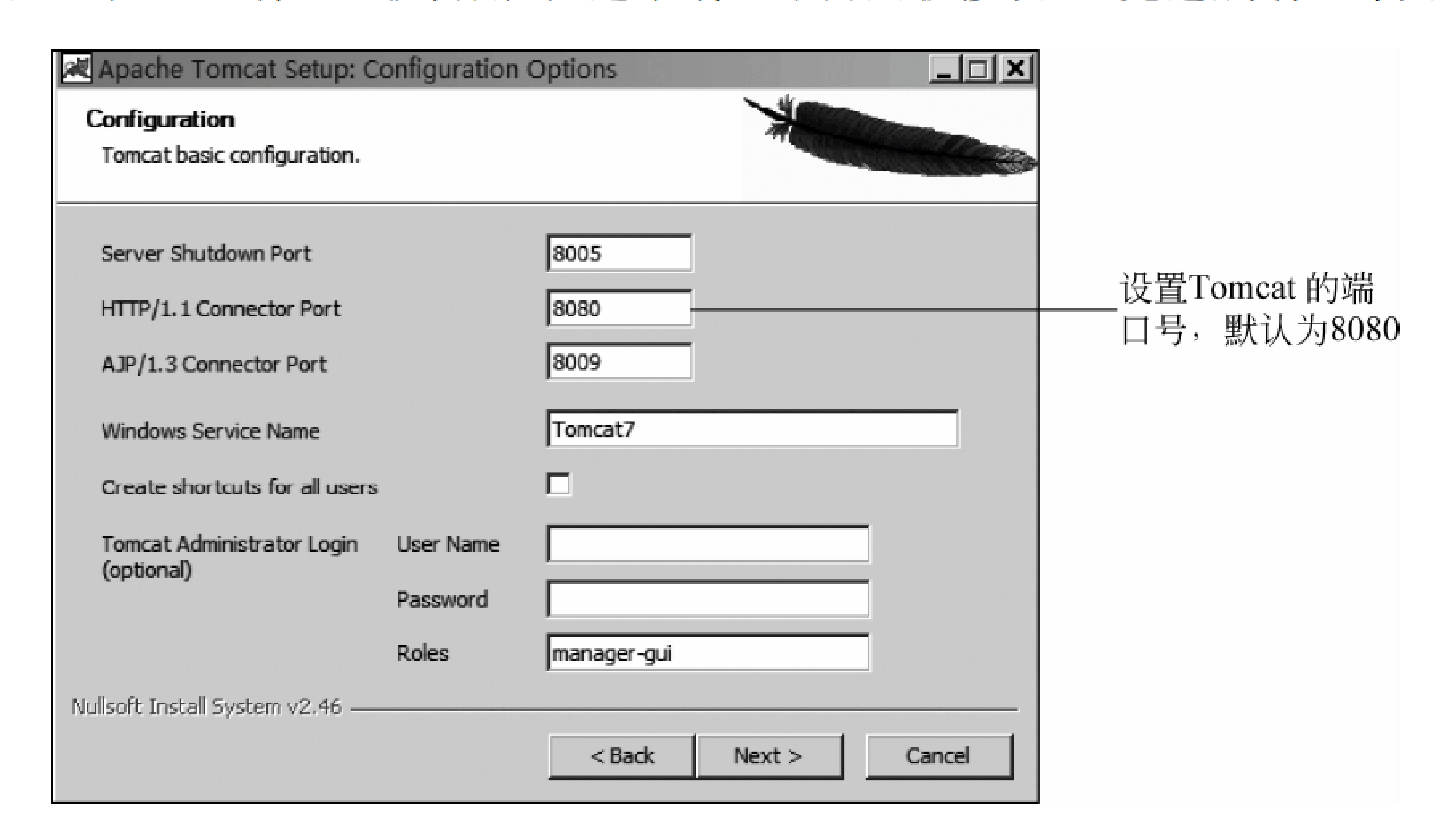


图 2-12　Tomcat 7.0 的安装

安装完成后，单击系统的"开始"菜单，找到 Apach Tomcat 7.0 选项，如图 2-13 所示，单击该选项下的 Configure Tomcat，弹出如图 2-14 所示的对话框。启动 Tomcat。

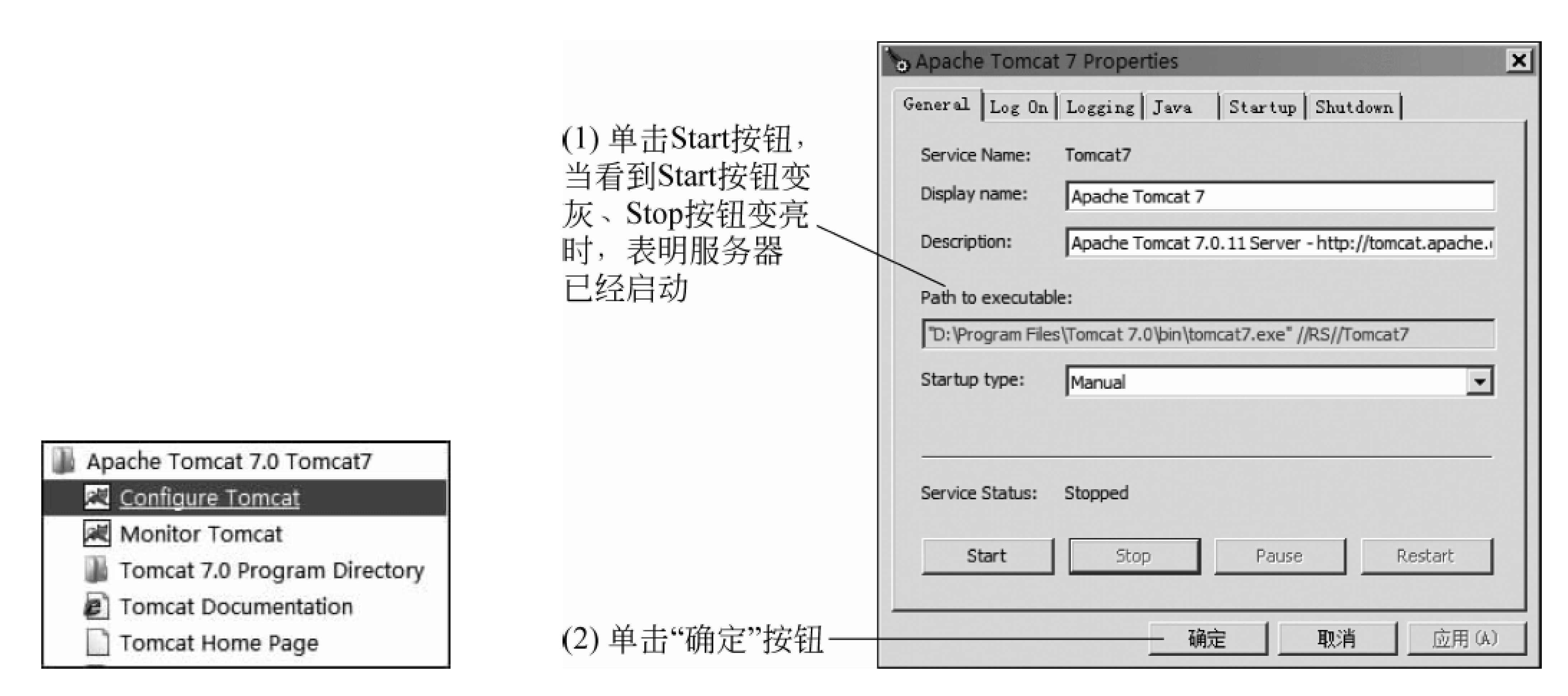


图 2-13　Tomcat 服务器的启动程序　　图 2-14　启动 Tomcat 服务器

待服务器正常启动后，打开 IE 或其他浏览器，在地址栏中输入 http://localhost:8080，如果看到如图 2-15 所示的页面，则表明 Tomcat 运行正常。

localhost 是本地主机的意思，表明服务器就是在当前使用的这台机器上。如果是进行远程的访问，可以用服务器的 IP 地址来替换 localhost；如果为服务器配置了域名，就可以通过域名来访问了。

在文件名默认的情况下，服务器会自动地寻找根目录下的 index.html、index.htm 或者 index.jsp 文件等默认的主页。

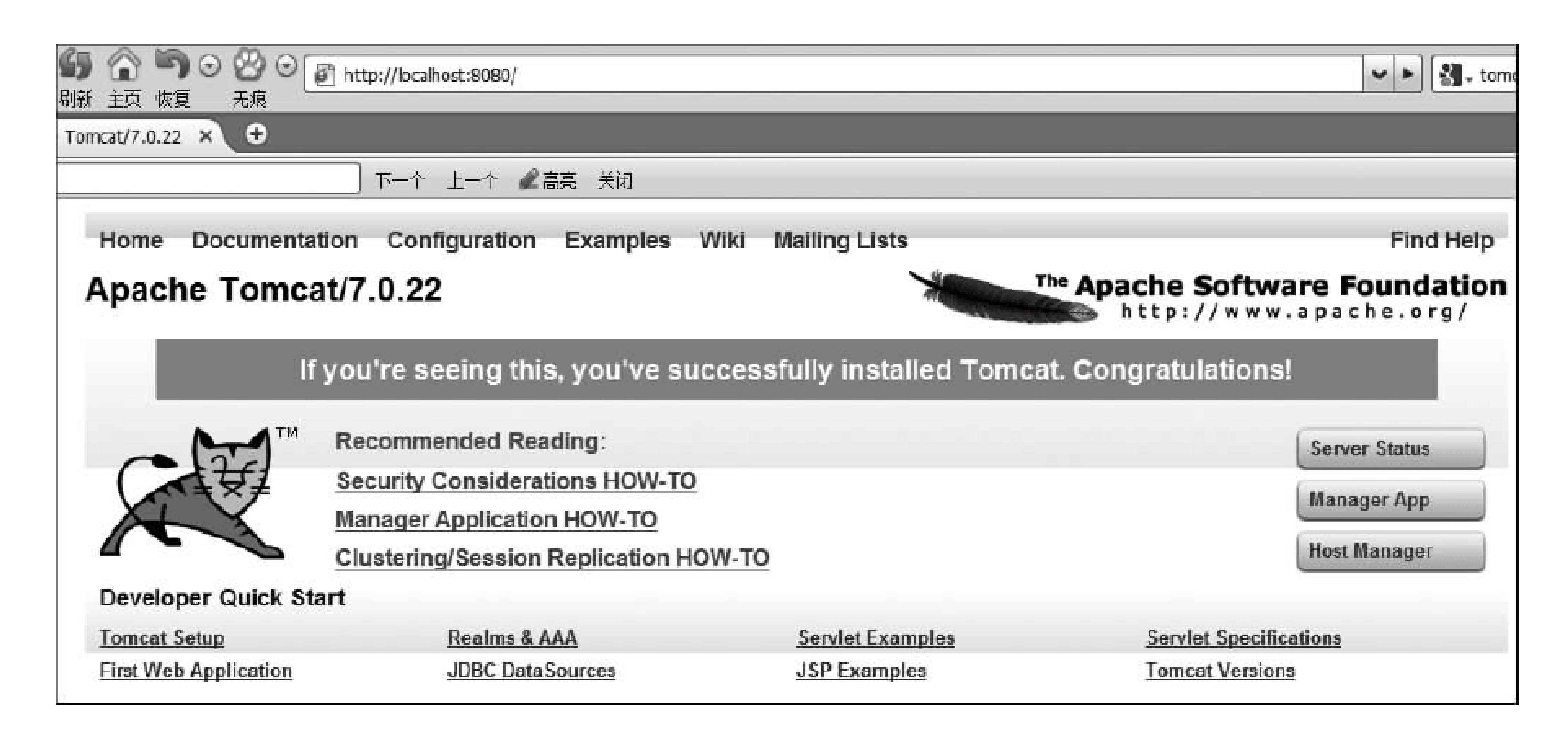


图 2-15　Tomcat 7.0 默认的首页

2.3　任务三：Web 项目的创建

MyEclipse 企业级工作平台是对 Eclipse IDE 的扩展，是目前最流行的 Java Web 集成开发环境，可以帮助项目开发人员极大地提高工作效率。MyEclipse 的安装较容易，按照安装向导进行默认安装即可，请读者自行下载安装。

下面将引导读者逐步熟悉 JSP+Java 的 Web 项目在 MyEclipse 9.0 中从开发到发布运行的操作流程。

任务目标：

在 MyEclipse 中，采用 JSP+Java 类的方式完成用户分组的判断。

当用户在浏览器的文本框中输入 1 时，在浏览器中输出“你好，普通用户！”；

当用户在浏览器的文本框中输入 2 时，在浏览器中输出“你好，集团用户！”；

当用户在浏览器的文本框中输入 3 时，在浏览器中输出“你好，VIP 用户！”。

技能训练：

MyEclipse 的使用。

下面开始创建一个 Java Web 项目。

第一步　选择工作空间

第一次进入 MyEclipse，会被要求选择工作空间（Workspace），即项目源代码所在的物理磁盘上的一个目录。一般情况下，不要把工作空间建在系统盘上。下面的例子中，工作空间被设置为 E 盘的 myprojs 目录，如图 2-16 所示。

第二步　创建 Web 项目

选择好工作空间后就会进入 MyEclipse 开发环境（首次进入 MyEclipse 会显示欢迎页，直接将它关闭就可以了），可以创建新的 Web 项目了，如图 2-17 所示。

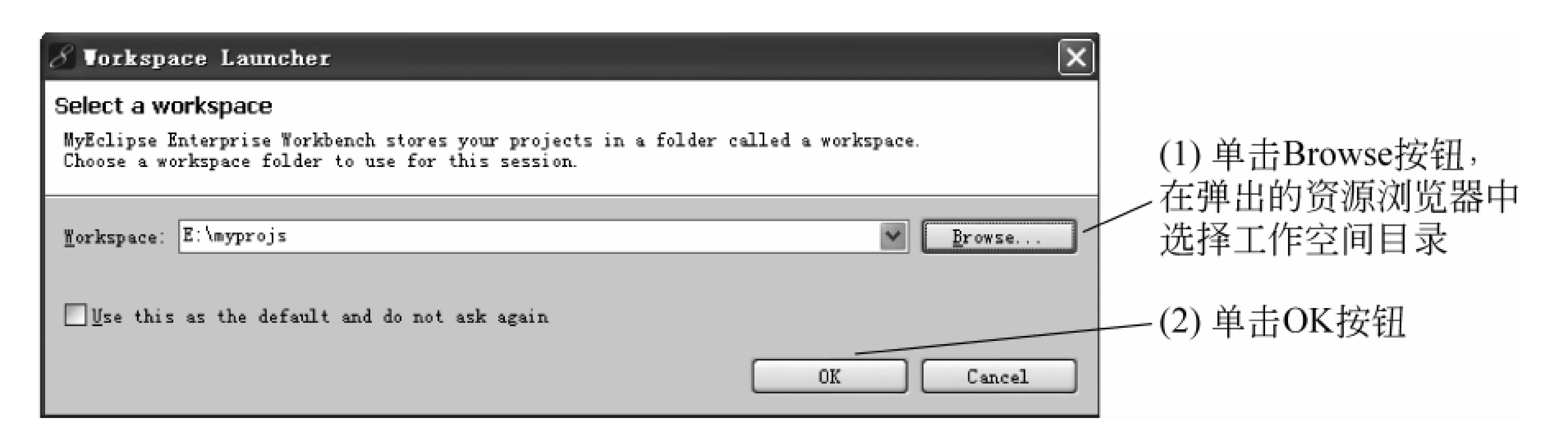


图 2-16　选择工作空间

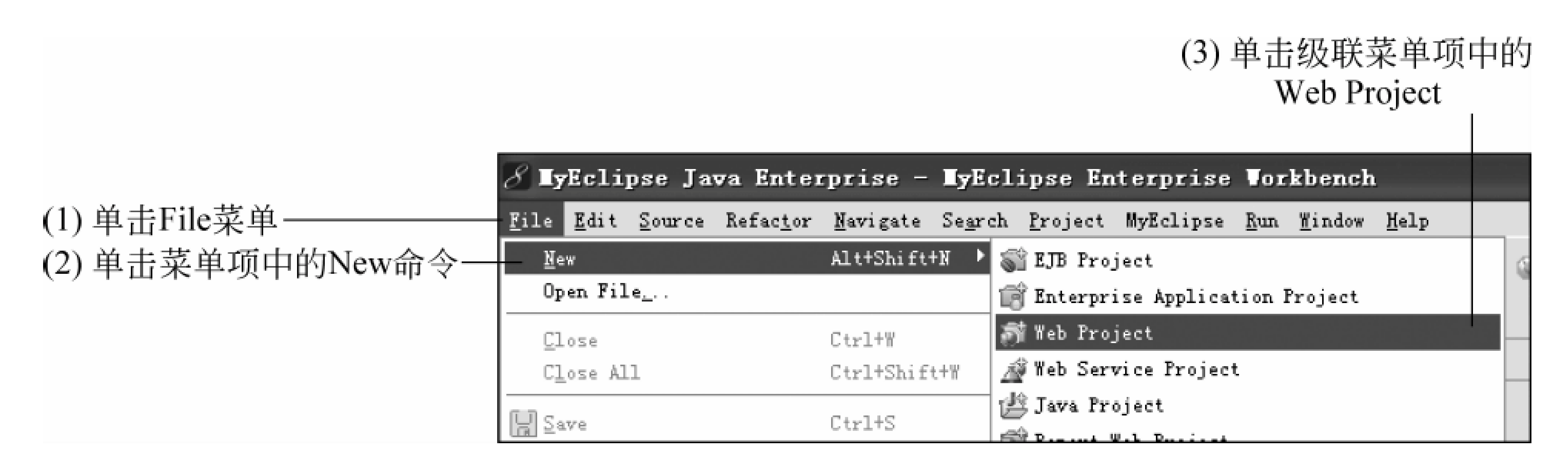


图 2-17　创建 Web 项目

在接下来出现的对话框中填写项目名称等信息，如图 2-18 所示。

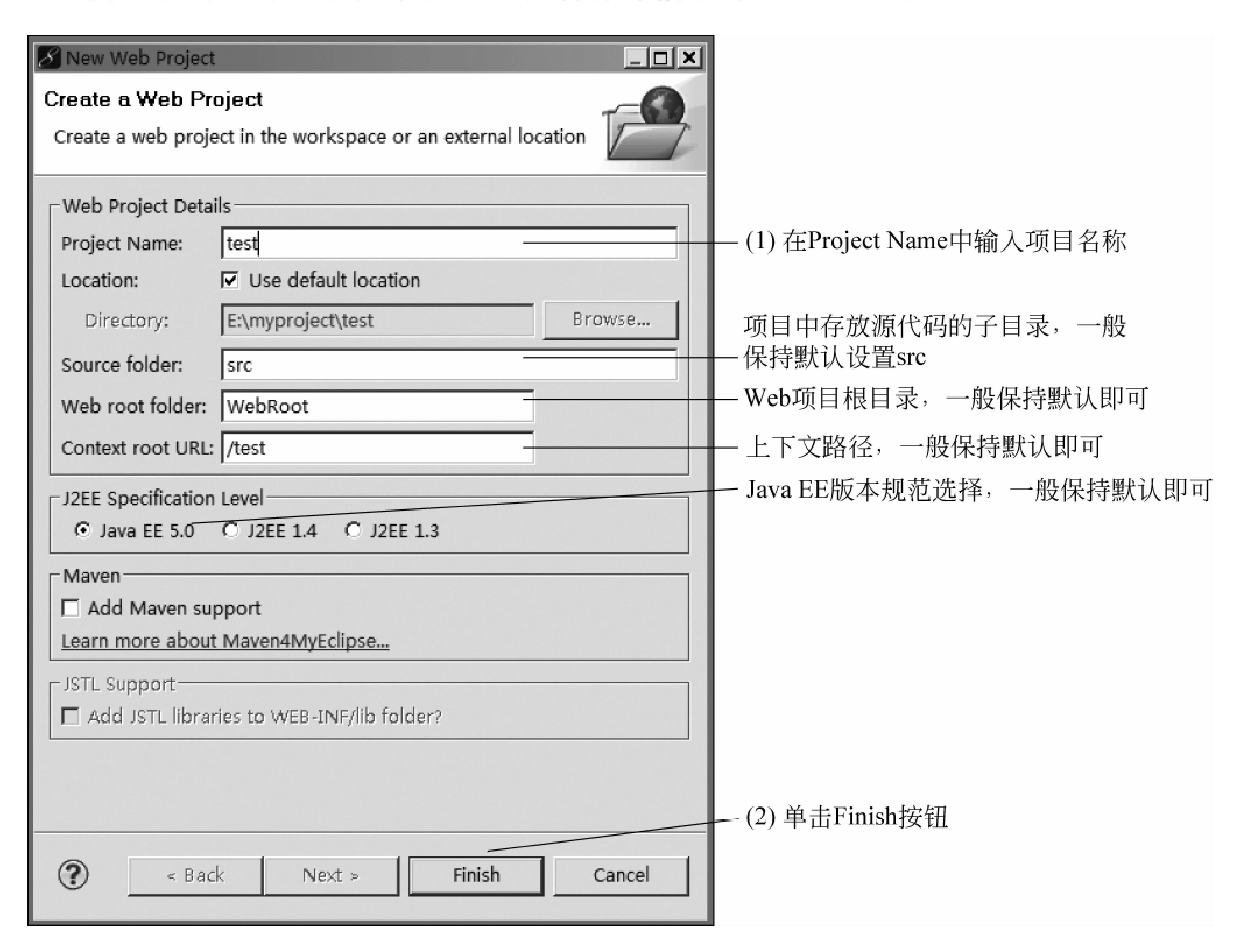


图 2-18　创建 Web 项目

第三步　新建包

项目中的 Java 类文件通常要用“包”来进行分类组织，所以在建立了一个新项目后，要为这个项目新建包，如图 2-19 所示。

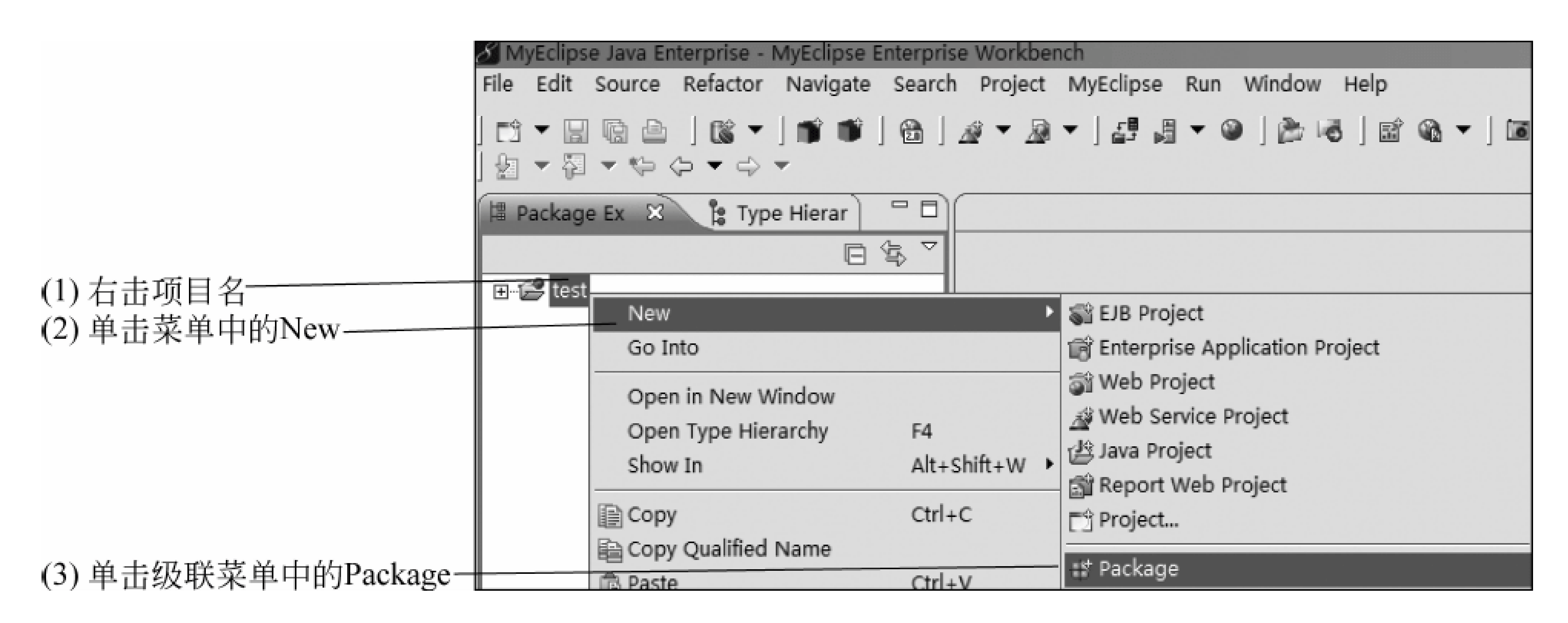


图 2-19　创建 Web 项目

在接下来出现的对话框中需要填写包的名称。Java 包的名字中的字母通常用小写，为了保障每个 Java 包命名的唯一性，要求在自定义的包的名称之前加上唯一的前缀。由于互联网上的域名称是不会重复的，所以一般采用互联网上的域名反序作为自定义包名的唯一前缀，如图 2-20 所示。

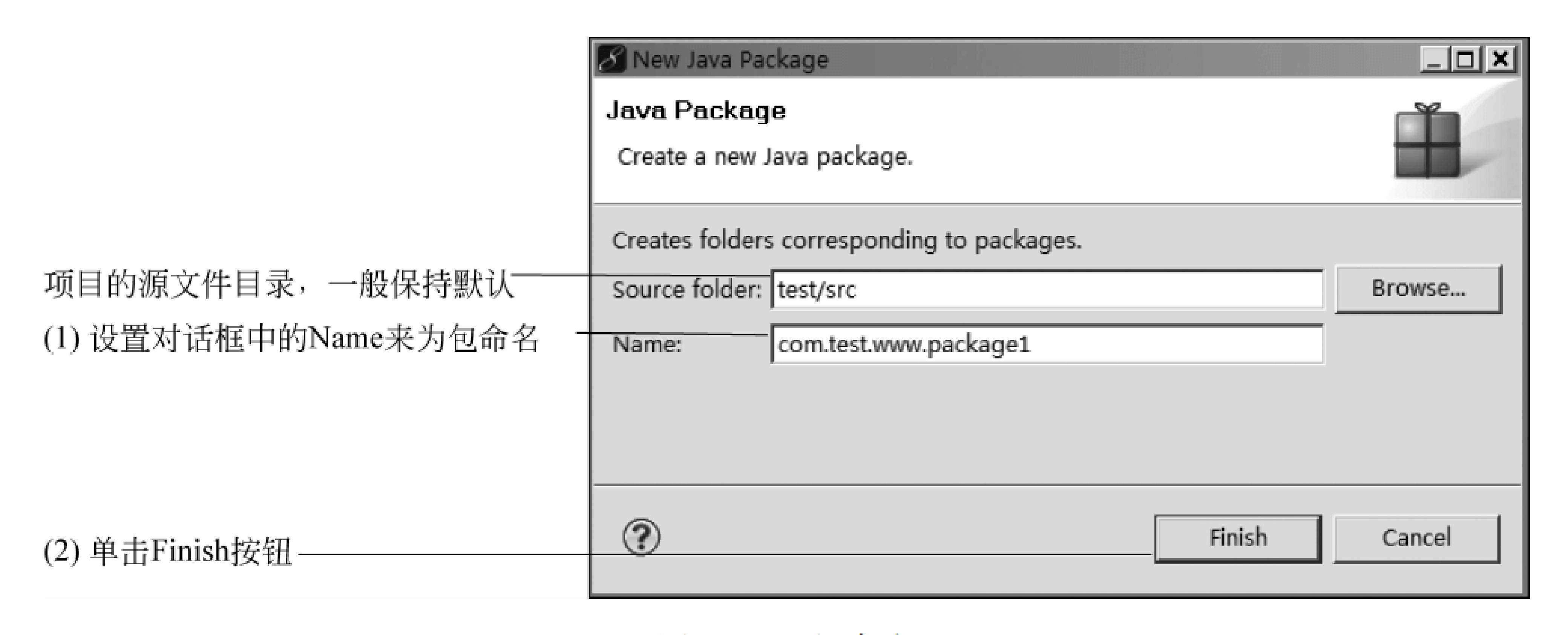


图 2-20　包命名

第四步　新建类

包被创建好之后，可以进一步在包中创建 Java 类，如图 2-21 所示。

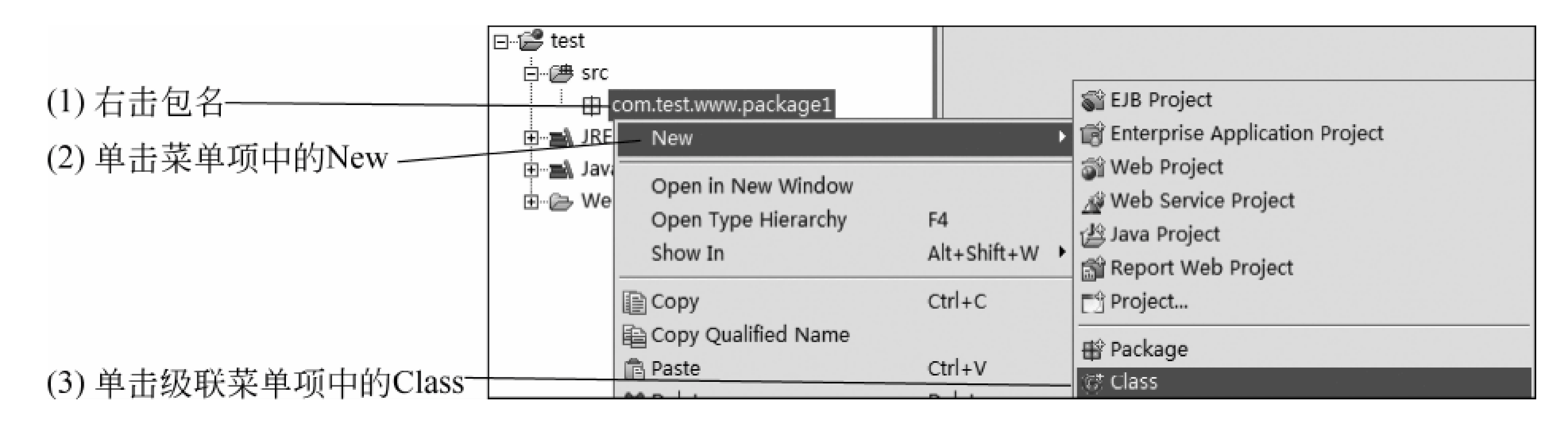


图 2-21　新建类

在接下来出现的 New Java Class 对话框中填写类的名称等信息，如图 2-22 所示。注意 Java 类的命名规范，应该以大写字母开头。

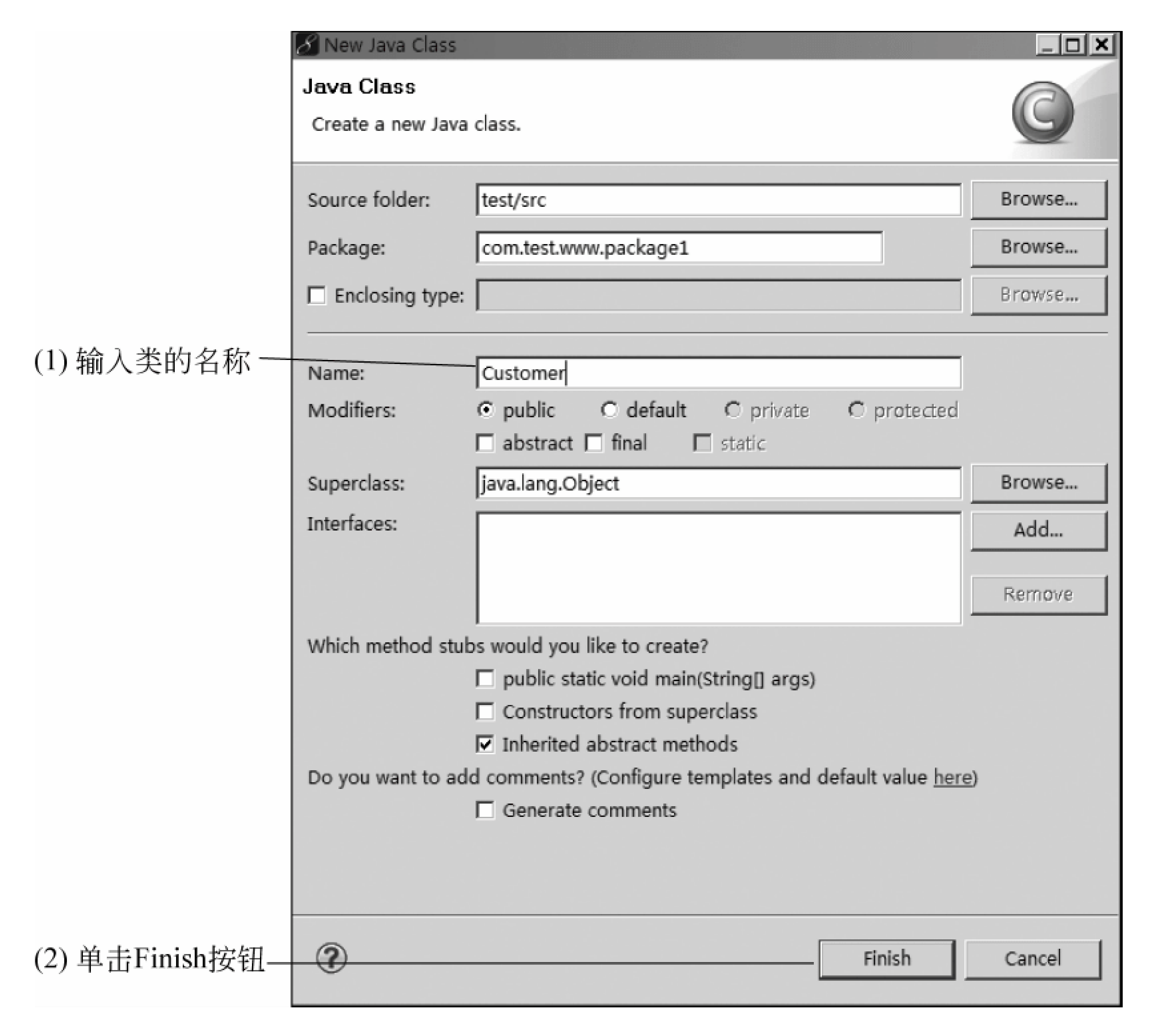


图 2-22　类命名

类 Customer 创建完成后，就会出现在包中。下面按照任务一的要求来编写 Customer.java 文件，完成客户级别的选择，如示例 2-1 所示。

示例 2-1　源文件：Customer.java

```
package com.test.www;

public class Customer
{
    public String getCustomerGroup(int group)
    {
        String WelcomeMessage="你好,访客";
        switch(group)
        {
            case 1: WelcomeMessage="你好,普通用户";
                    break;
            case 2: WelcomeMessage="你好,集团用户";
                    break;
            case 3: WelcomeMessage="你好,VIP 用户";
                    break;
        }
        return WelcomeMessage;
```

```
        }
    }
```

第五步 新建 JSP 文件

要完成任务一的要求，还需要编写 JSP 文件，以便在浏览器中进行显示。下面新建一个 JSP 文件，创建的步骤如图 2-23 所示。

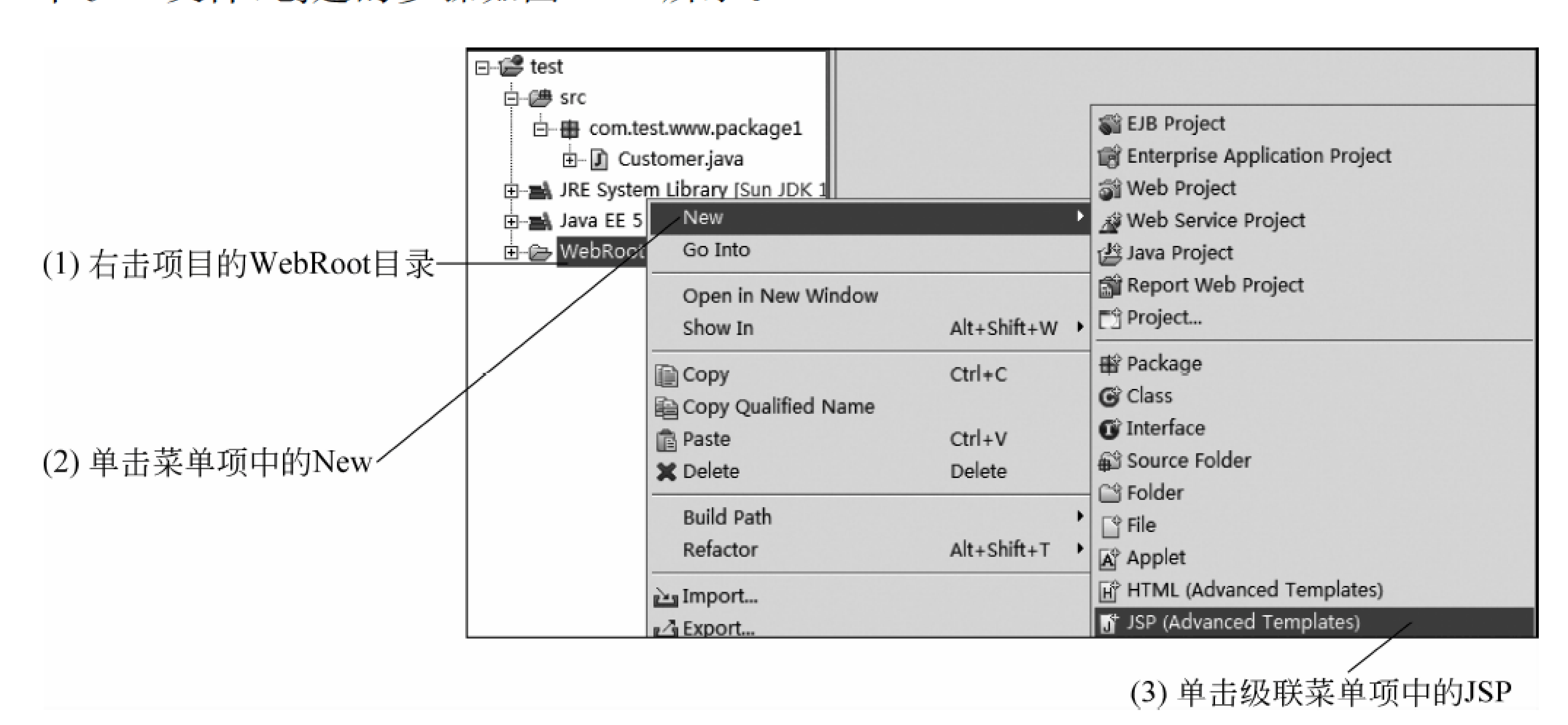


图 2-23 新建 JSP 文件

在接下来出现的 Create a new JSP page 对话框中输入 JSP 文件的名称，如图 2-24 所示。

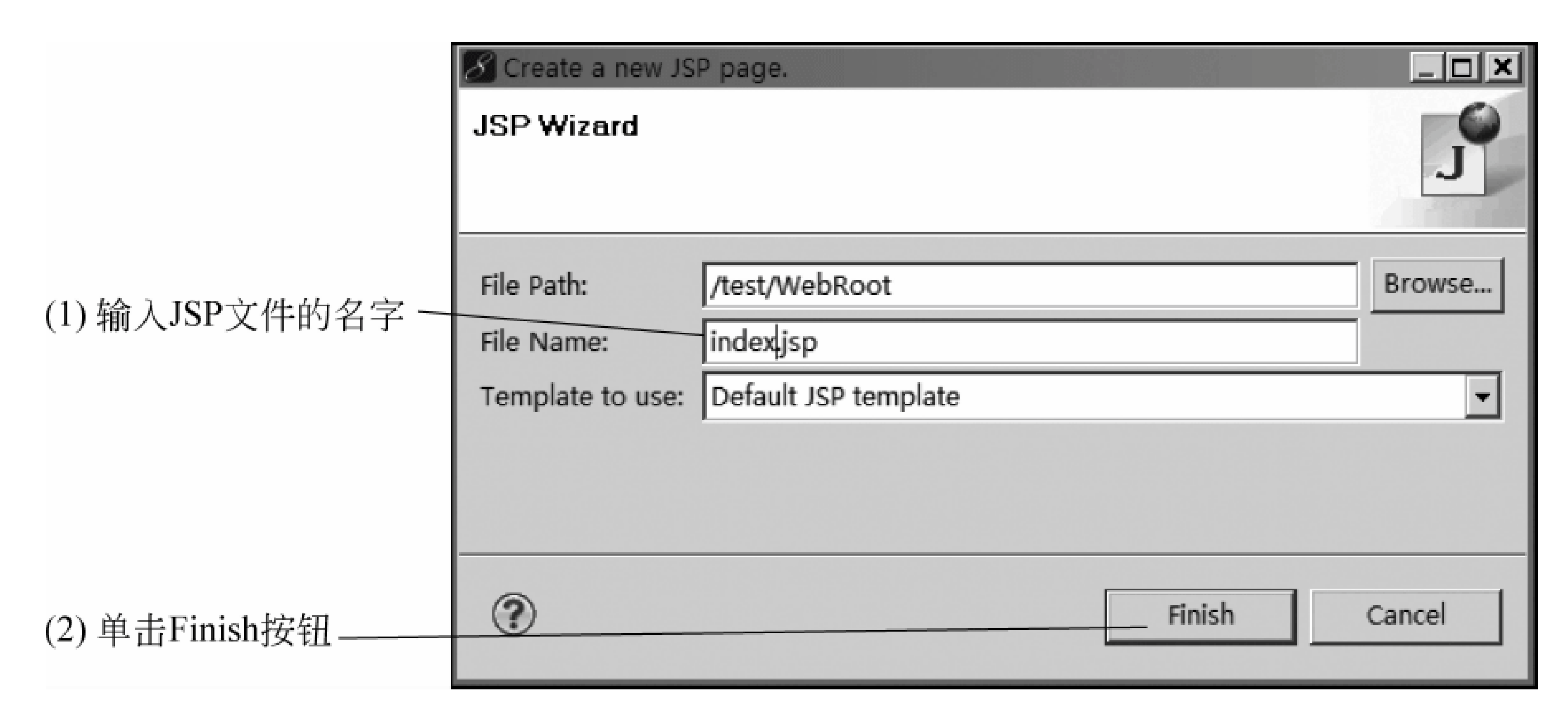


图 2-24 命名 JSP 文件

创建完 index.jsp 之后，编写代码完成用户输入信息的获取，并调用示例 2-1 中类的方法，完成用户组别的判断，再将结果显示输出。如示例 2-2 所示。

示例 2-2 源文件：index.jsp

```
<%@page language="java"  pageEncoding="GB18030"
                import="com.test.www.package1.*,java.util.*" %>
```

引入Java类所在的包

```
<form name="form" method="post" action="">
        请输入客户类别  1：普通用户  2：集团用户 3：VIP 用户
        <input type="text" name="group" />
</form>
    <%
        String groupS=request.getParameter("group");
        try
        {
          int group=Integer.parseInt(groupS);
          Customer customer=new Customer();      创建Customer类的实例
          String mess=customer.getCustomerGroup(group);
          out.print(mess);                       调用实例的getCustomerGroup方法
        }catch()
        {
        out.print("请输入 1~3 之间的一个整数");
        }
        %>
```

2.4 任务四：在 MyEclipse 中配置 Web 服务器

到此为止，一个 Java Web 项目已经建立完成了。但是如果要运行，还需要 Web 服务器的支持。本案例选用 Tomcat 7.0 作为 Web 服务器。Tomcat 是一个应用较广泛且为开源的 Web 服务器，其安装过程比较简单，在此不做详细介绍，请读者自行下载并安装。

服务器安装好之后，可以在操作系统中启动并完成相应的功能。MyEclipse 中也集成了服务器的设置和启动等功能，使项目的开发和调试更加方便、快捷。

任务目标：

为 MyEclipse 配置 Tomcat 7.0 服务器，配置完成后，在 MyEclispe 中启动服务器，使之可以正常运行。

技能训练：

MyEclipse 的使用。

下面以图示的方式，逐步介绍如何在 MyEclipse 中配置和启动 Web 服务器。

第一步 配置 Tomcat 7.0 服务器

在 MyEclipse 菜单下方的功能按钮中找到 Run/Stop/Restart MyEclipse Servers 按钮，单击右侧的向下箭头，然后再单击菜单项中的 Configure Server，如图 2-25 和图 2-26 所示。

在图 2-26 所示菜单项中单击 Configure Server，会弹出一个对话框，在此对话框中配置 Tomcat 7.0 服务器的步骤如图 2-27 所示。

第二步 启动 Tomcat 7.0 服务器

完成第一步的配置后，Tomcat 7.x 会出现在 MyEclipse 可用服务器列表中，可以方便地启动服务器，如图 2-28 所示。

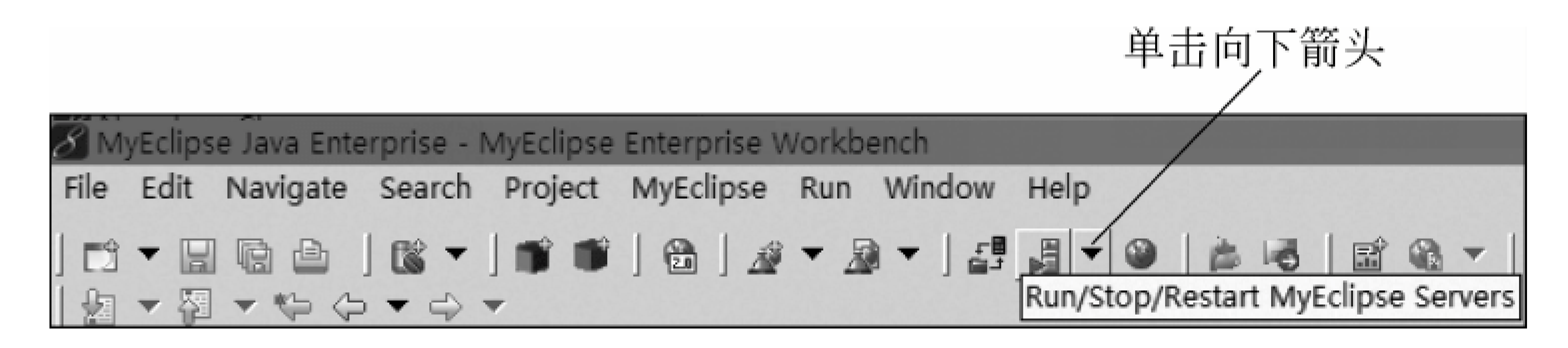


图 2-25　MyEclipse 功能按钮



图 2-26　MyEclipse 服务器按钮的下拉菜单项

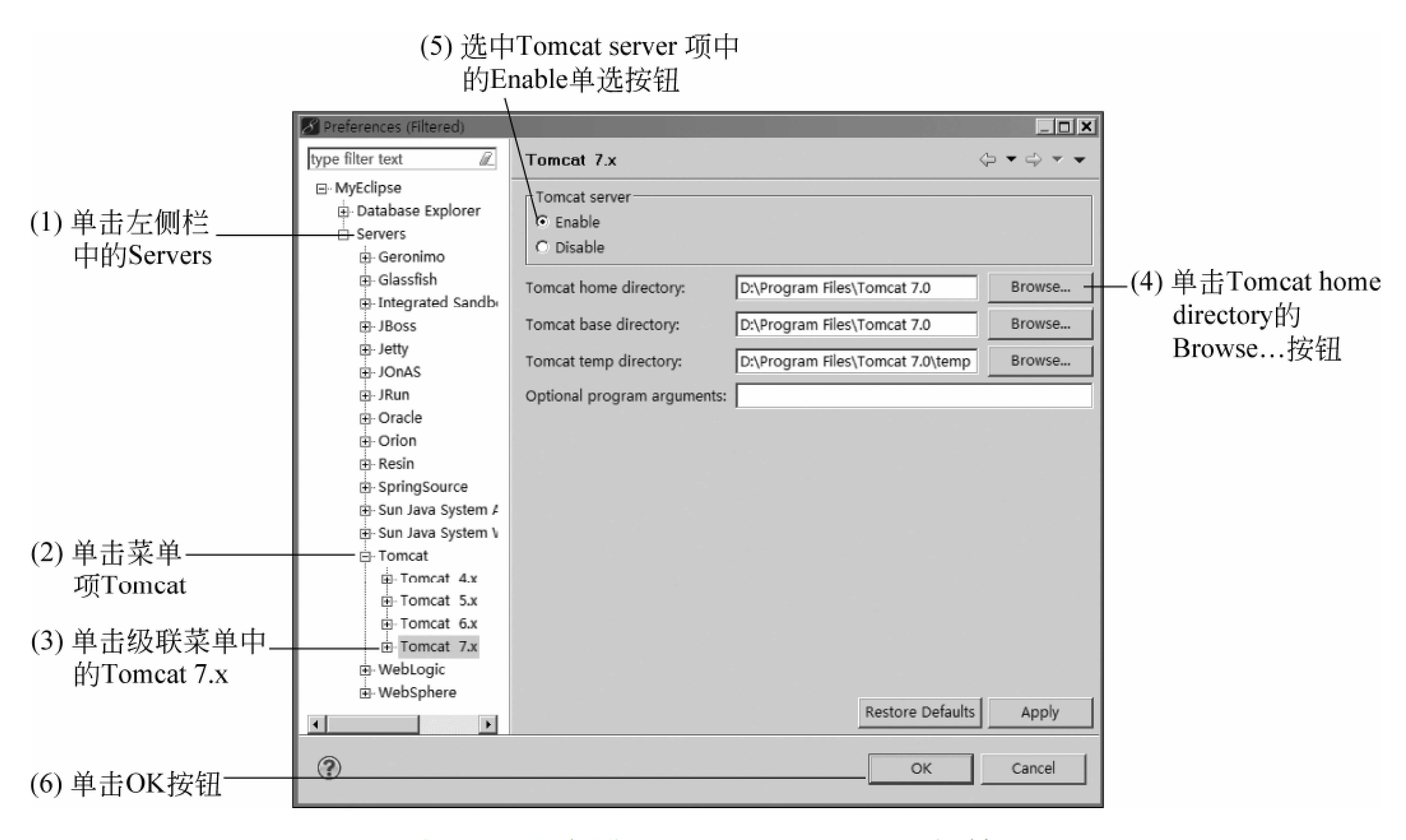


图 2-27　服务器 Preferences(Filtered)对话框

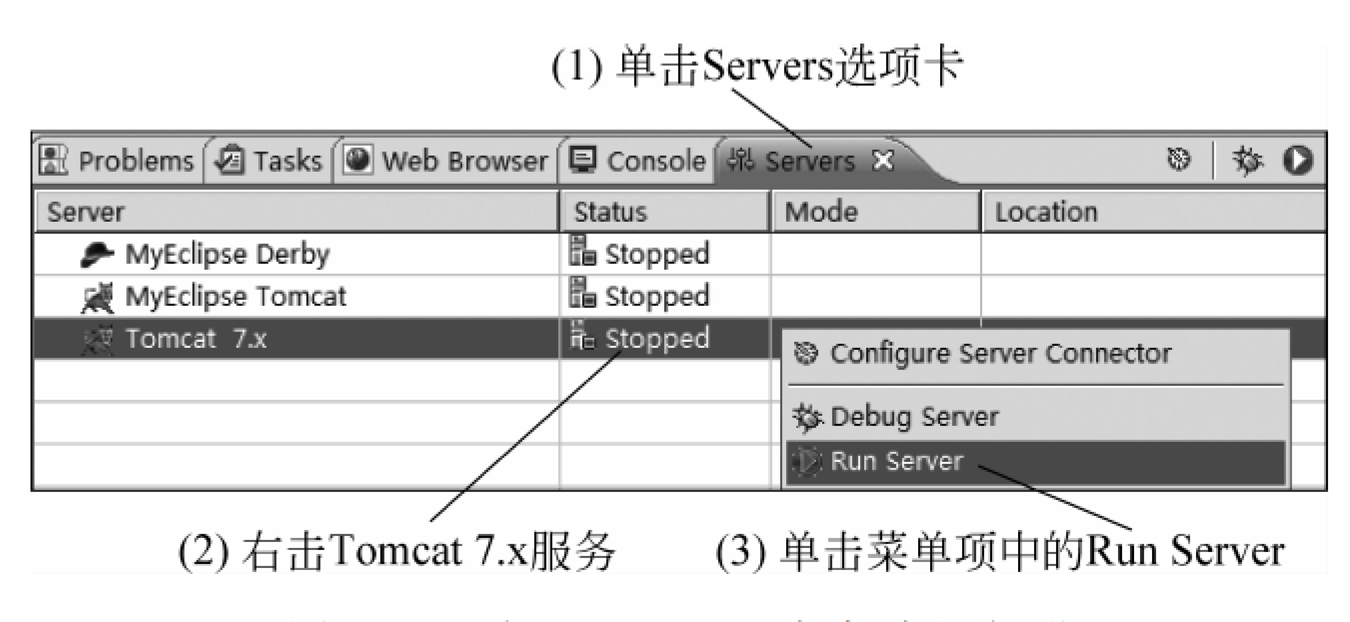


图 2-28　在 MyEclipse 中启动服务器

启动服务器后，在 Console 选项卡中会出现服务器启动的相关信息，如果在此过程中出错，Console 中会出现错误提示；如果正常，服务器状态将由 Stopped 变为 Running。

2.5　任务五：向 Web 服务器上部署项目

服务器正常启动后，在浏览器的网页地址栏中输入 http://localhost:8080，就可以看到 Tomcat 7.0 的默认主页了。但是现在还不能够访问本章任务一编写的 JSP 文件，要使任务一中完成的项目运行起来，还需要将项目部署到服务器上。下面以图示的方式引导读者完成项目的部署。

第一步　设置项目发布的目标目录

项目部署到服务器上的目标目录可以由开发人员通过设置项目属性来指定，如图 2-29 和图 2-30 所示。

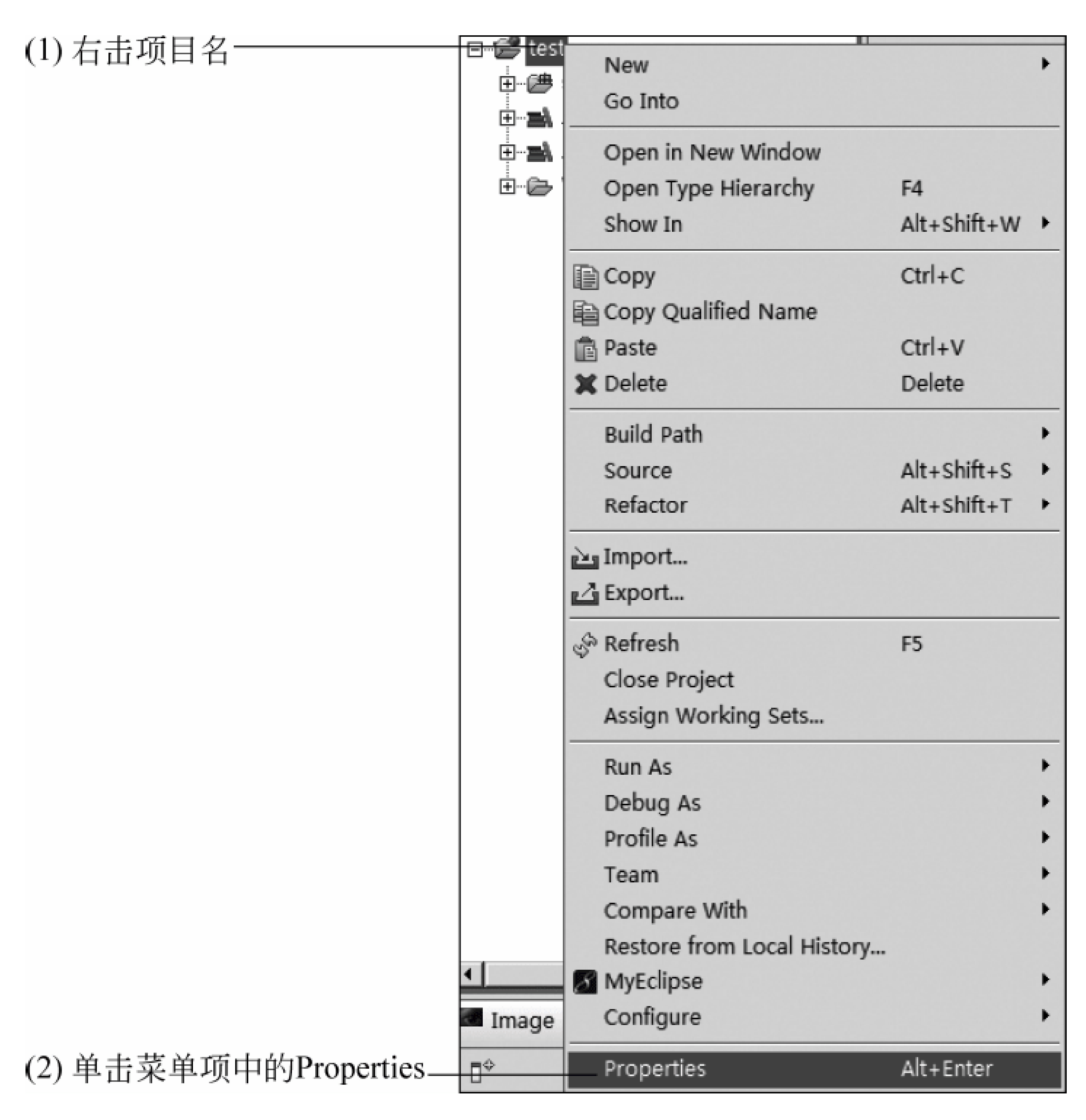


图 2-29　设置项目属性

第二步　设置项目部署的目标目录

如图 2-31 和图 3-32 所示是向服务器上部署项目的向导图示。

单击 Add Deployment 命令后会弹出一个对话框，如图 2-32 所示。

项目部署成功后，会出现在服务器下方，状态为 OK，如图 2-33 所示。

本章的三个任务都完成后，就可以在 IE 浏览器中访问项目的 index.jsp，从而完成用户组的判断了。

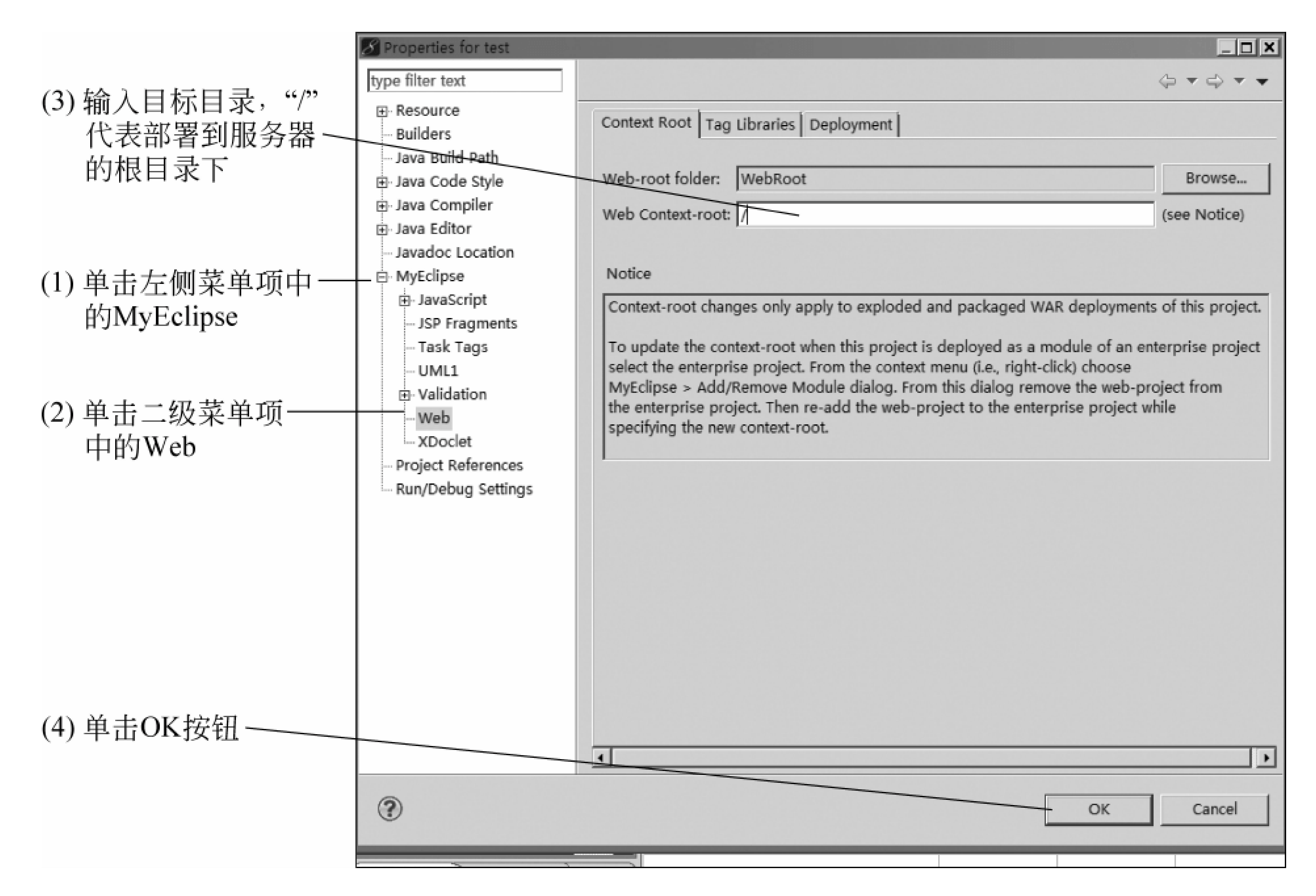


图 2-30　设置项目属性

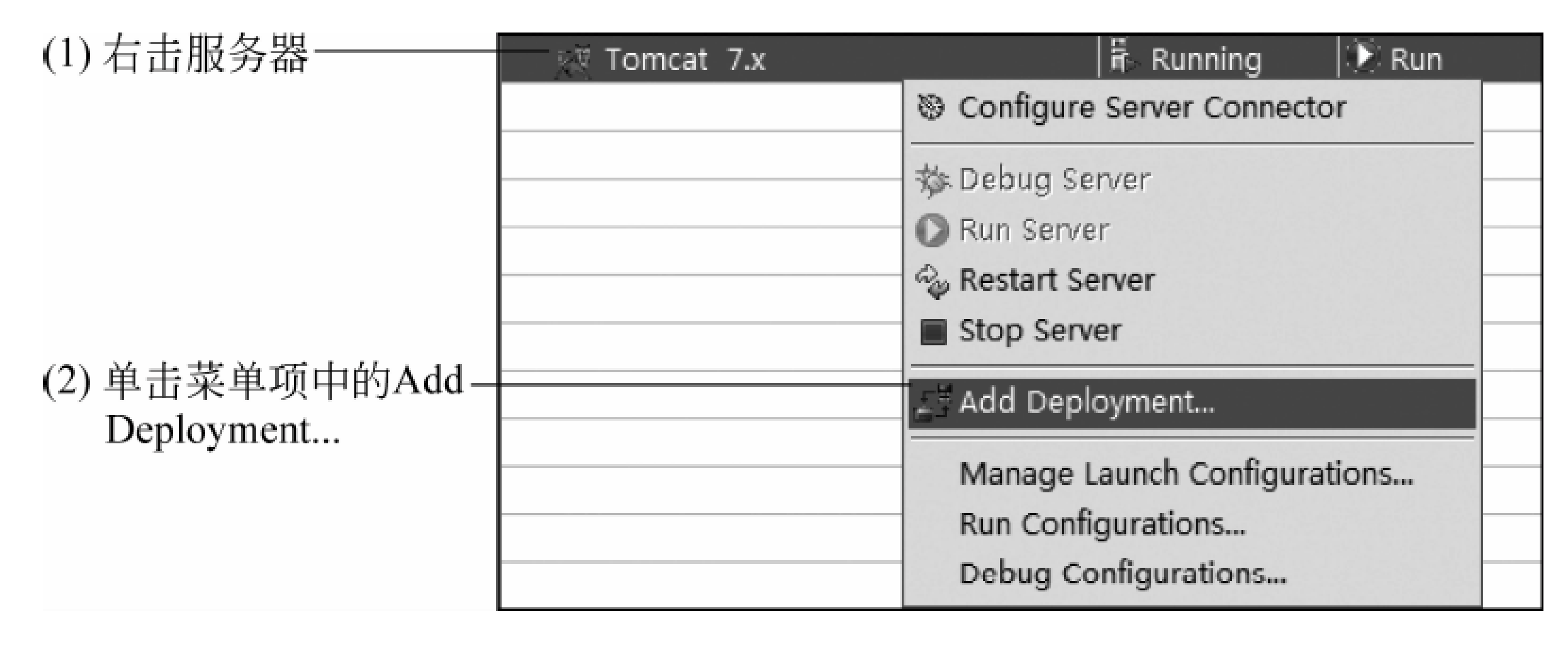


图 2-31　设置项目属性

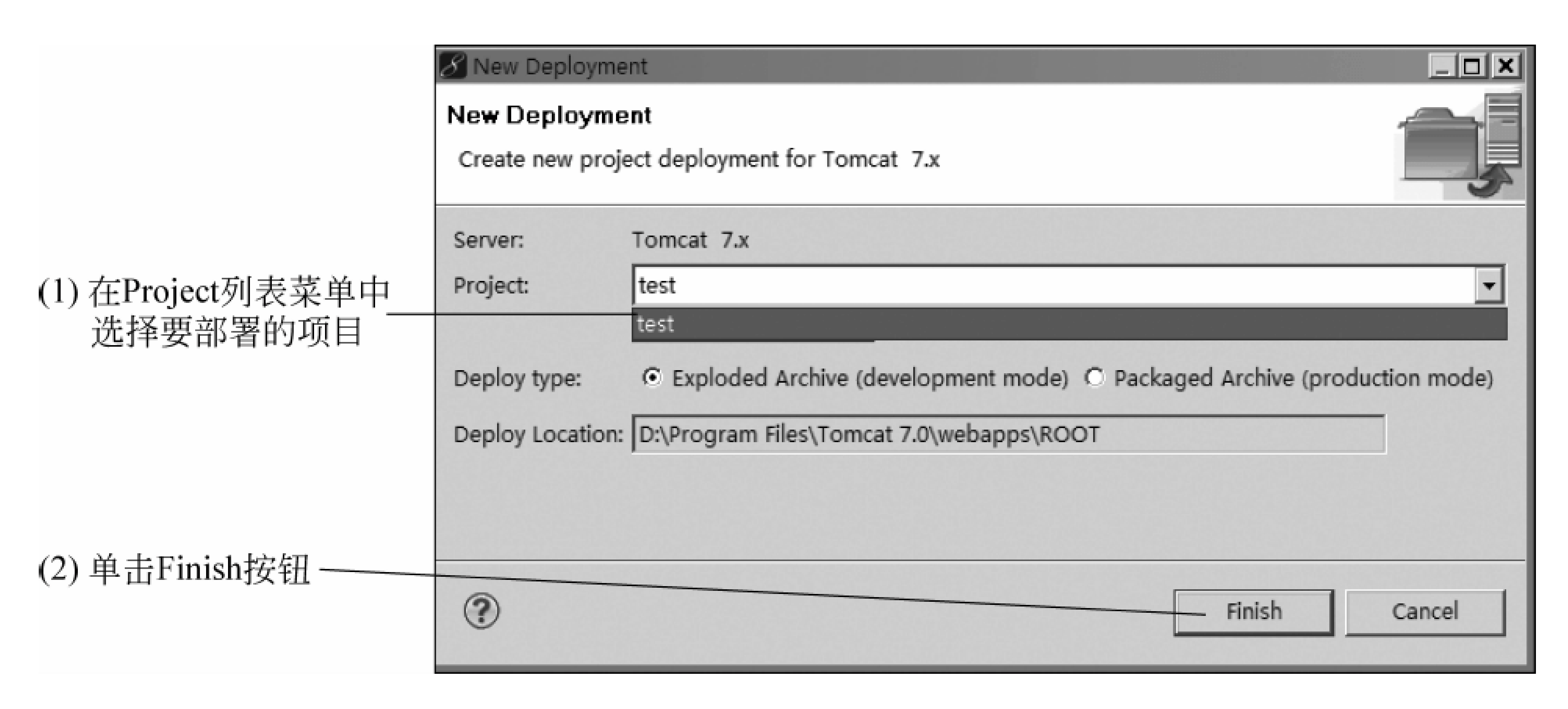


图 2-32　选择要部署的项目



图 2-33　项目部署成功

课 后 练 习

1. 请读者下载 Tomcat 6.0 和 MyEclipse 6.0 以上版本并完成安装。

2. 采用 JSP+Java 类的方式完成累加计算：根据用户输入的数字 n，在浏览器中显示 1～n 累加的值。

要求：在 MyEclipse 中创建一个 Java Web 项目，实现题目的要求，并成功部署到服务器上。

3. 采用 JSP+Java 类的方式完成对用户输入的两个浮点数的四则运算：用户输入两个浮点数，并选择四则运算中的一种，在浏览器中显示运算结果。

要求：在 MyEclipse 中创建一个 Java Web 项目，实现题目的要求，并成功部署到服务器上。

提示：字符串转浮点的方法：float f=Float.parseFloat(String s)。

4. 在 MyEclipse 中分别为“佳衣屋”和第 1 章中的“自选项目”创建一个 Web Project，为后续章节做好准备工作。

第 3 章　MySQL 数据库的设计与开发

本章学习要点：

- 熟练掌握在 SQLyog 9.0 中设计与实现 MySQL 数据库的方法；
- 熟练掌握数据库导入与备份的方法；
- 熟练掌握数据库的连接池技术。

数据库的设计与开发是 Web 项目开发不可缺少的一个环节，Web 应用中的数据都存储在数据库中。本书中的项目开发采用了 MySQL 这一开源数据库，版本为 5.1，同时采用 SQLyog 9.0 作为 MySQL 数据库的开发管理工具，请读者自行下载并安装 MySQL 和 SQLyog 9.0。本章介绍如何使用 SQLyog 9.0 来开发项目的数据库。

3.1　任务一：创建数据库

任务目标：

使用 SQLyog 为"佳衣屋"项目创建 MySQL 数据库。

技能训练：

SQLyog 的用法。

MySQL 和 SQLyog 都安装完毕后，首先打开 SQLyog 与 MySQL 服务器建立连接，如图 3-1 所示。

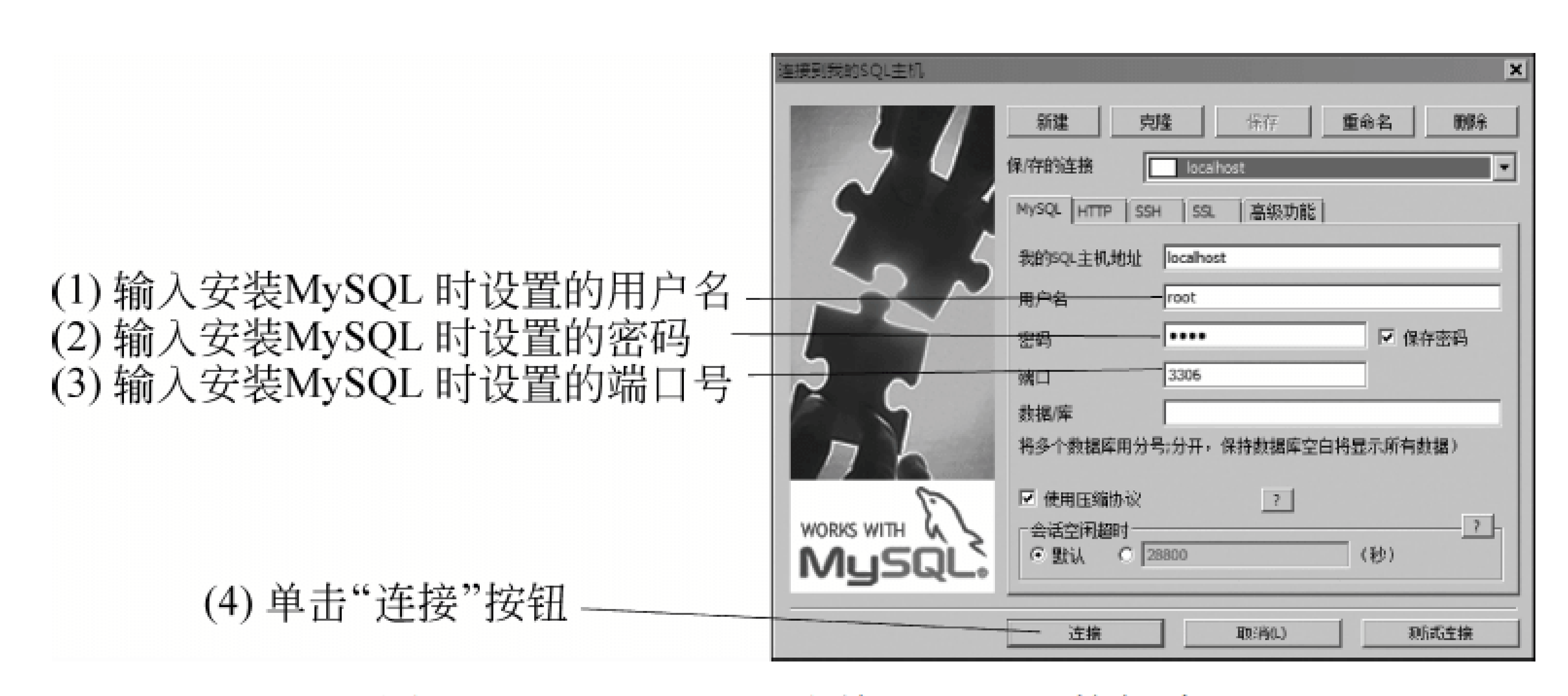


图 3-1　用 SQLyog 连接 MySQL 数据库

连接上 MySQL 服务器后，就可以创建数据库，如图 3-2 和图 3-3 所示。

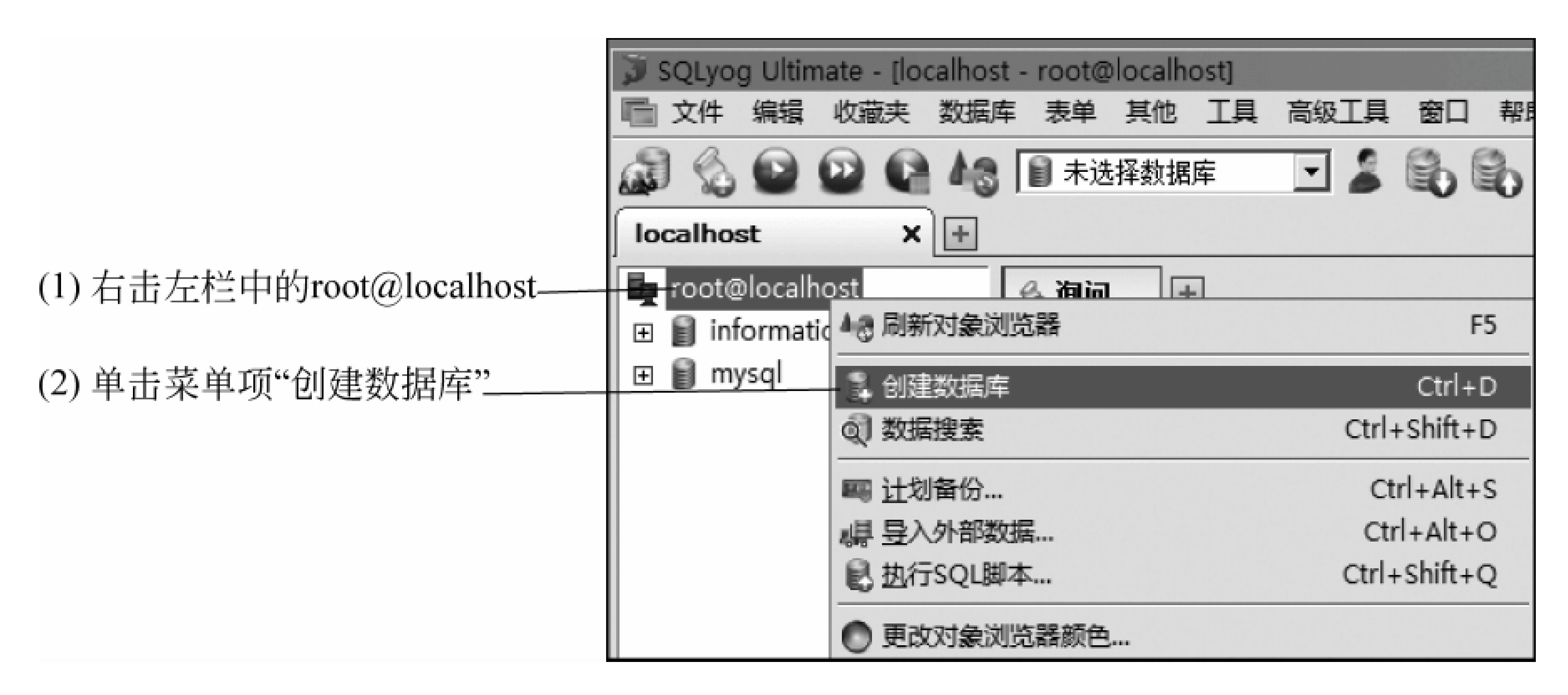


图 3-2　创建数据库

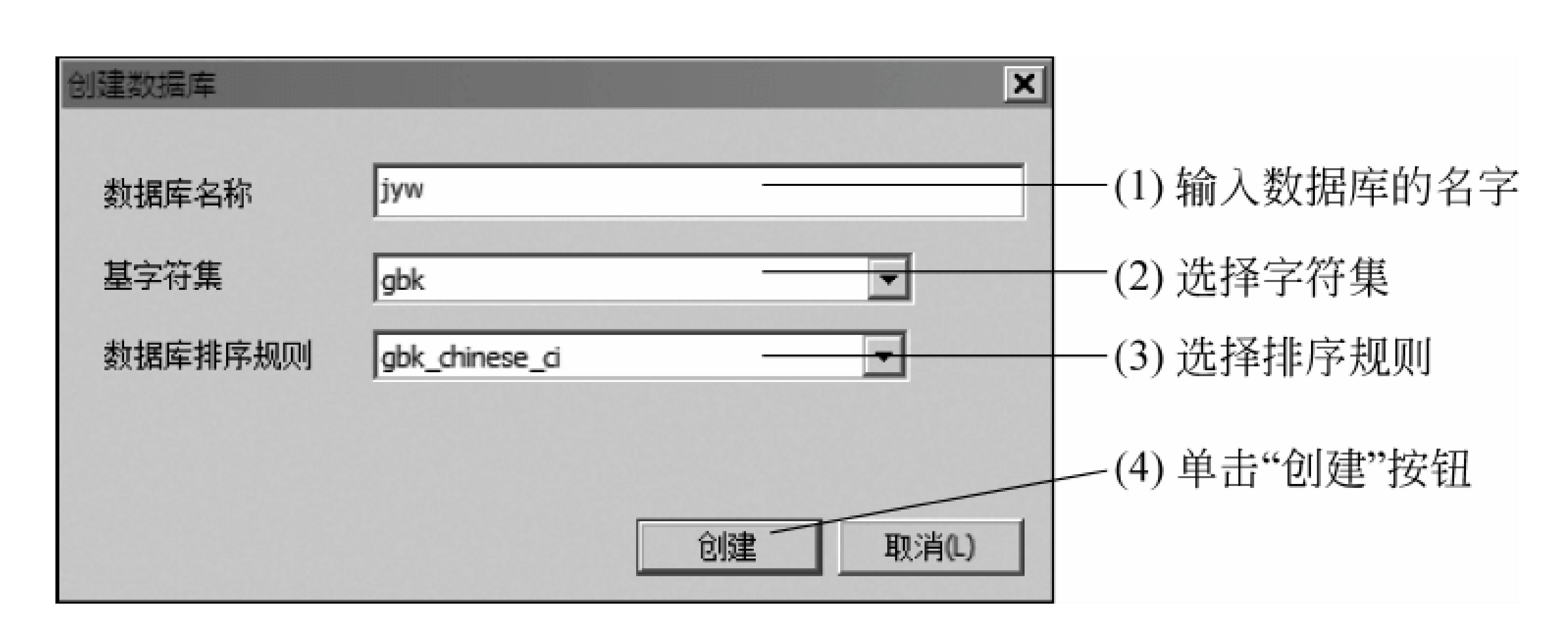


图 3-3　新数据库的基本配置

单击图 3-2 中的“创建数据库”菜单项后，SQLyog 弹出一个对话框，要求对新数据库做一个简单的配置，如图 3-3 所示。

单击图 3-3 中的“创建”按钮后，数据库 jyw 就创建好了。

3.2　任务二：创建数据表

任务目标：

使用 SQLyog 为任务一中创建的数据库 jyw 创建存储商品信息的数据表 goods。

技能训练：

SQLyog 的用法。

创建新表（见图 3-4）需要为表命名，并进行创建列（即字段）等的设计，如图 3-5 所示。

按照图中的步骤操作完成后，数据表 goods 就创建完成了。本书后续章节任务示例中需要的数据表的设计如表 3-1～表 3-5 所示，请读者自行完成这些数据表的创建。

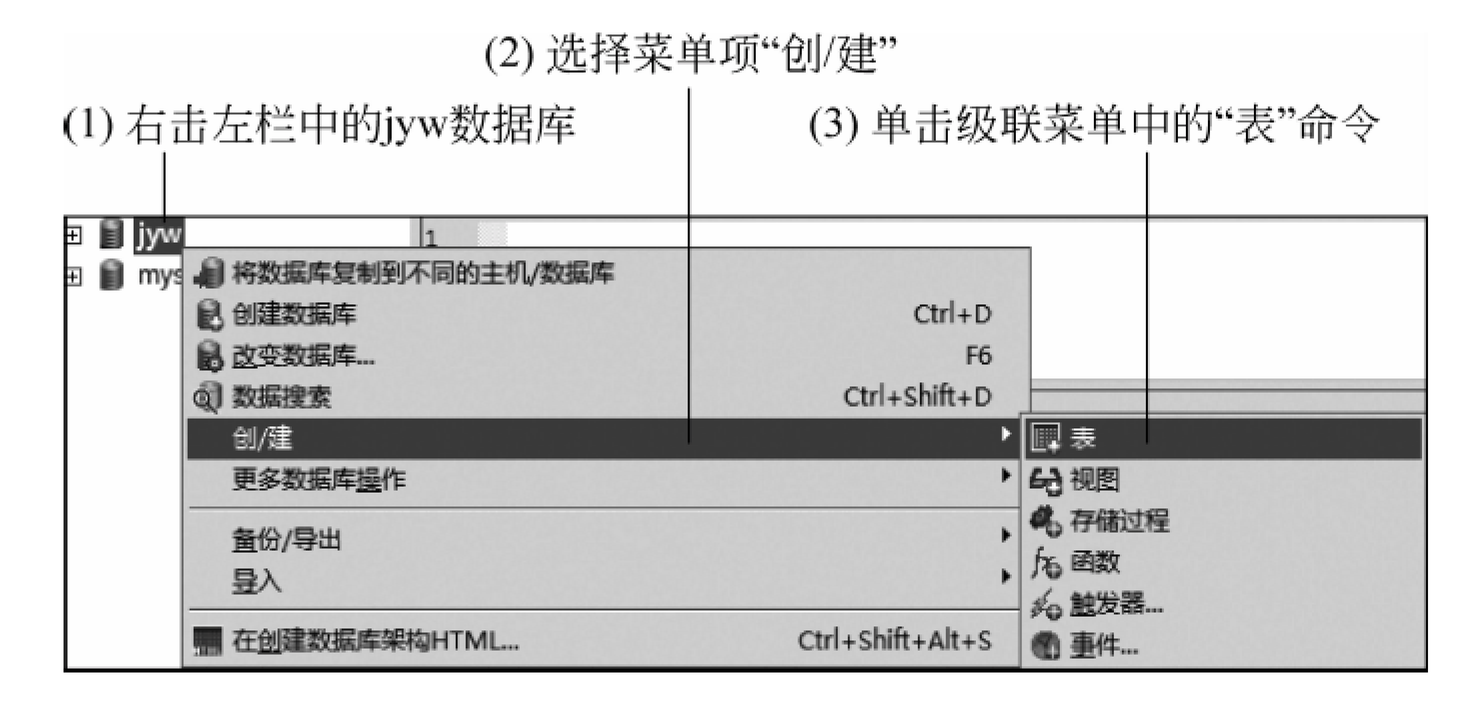


图 3-4　创建表

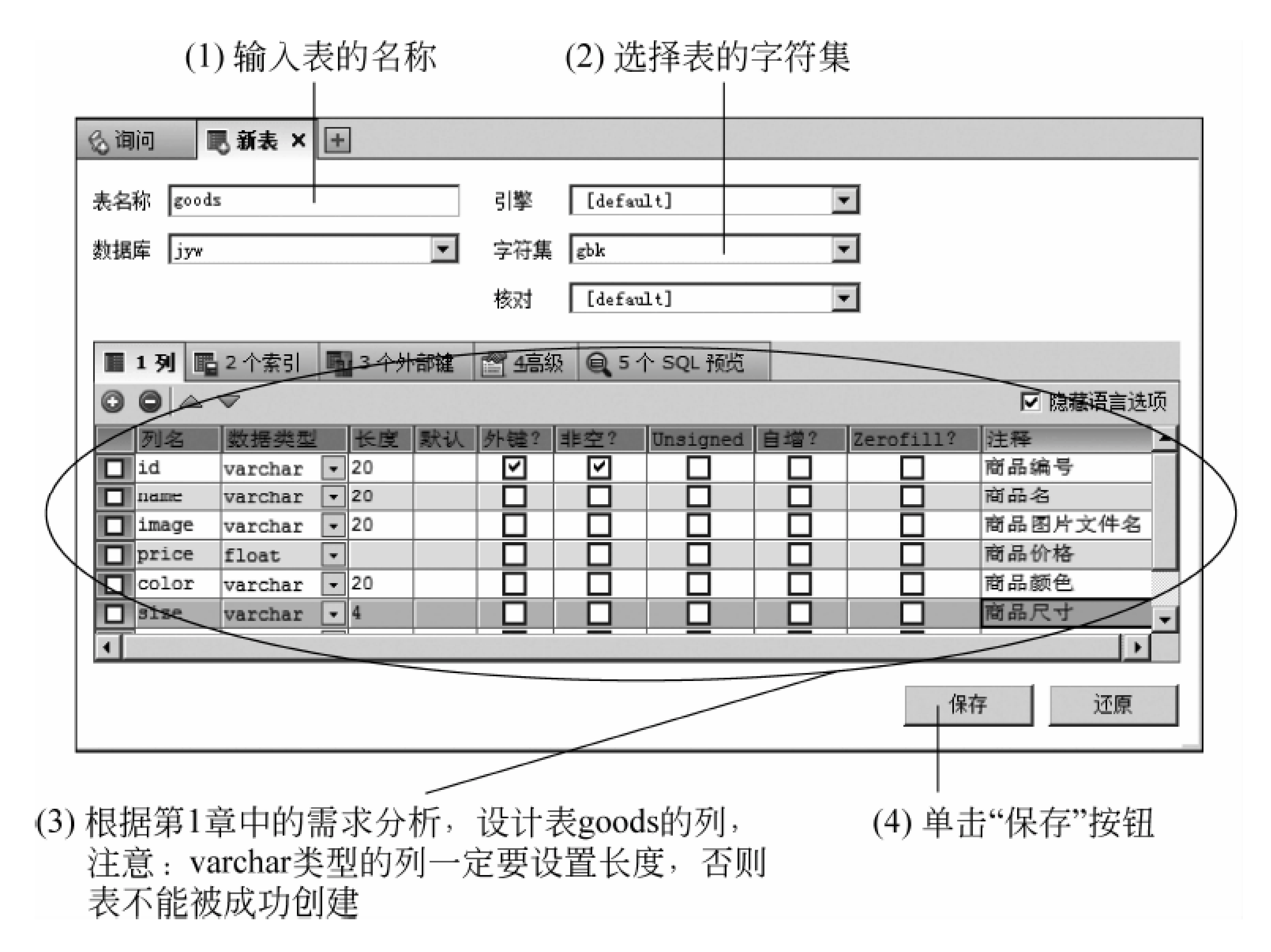


图 3-5　数据表 goods 的设计

表 3-1　员工信息数据表 employee

列名称	数据类型	长度	说　明
id	int		序号、主键、自动增长
account	varchar	20	用户名
password	varchar	8	密码
storenum	int		所属分店
name	varchar	10	真实姓名
authority	int	1	权限

表 3-2　销售信息数据表 sales

列名称	数据类型	长度	说　明
id	int		序号、主键、自动增长
goodsid	varchar	20	商品编号
discount	int		折扣

续表

列名称	数据类型	长度	说　　明
date	timestamp		销售时间
quantity	int		销售数量
salesperson	varchar	10	销售人员
size	varchar	5	尺码
storenum	int		分店编号

表 3-3　库存信息数据表 stock

列名称	数据类型	长度	说　　明
id	int		序号、主键、自动增长
stock	int		库存量
size	varchar	5	尺码
goodsid	varchar	20	商品编号
storenum	int		分店编号

表 3-4　门店信息数据表 storeinfo

列名称	数据类型	长度	说　　明
id	int		序号、主键、自动增长
name	varchar	50	分店名称
addr	varchar	70	分店地址
contact	varchar	50	分店联系方式

表 3-5　进货信息数据表 orders

列名称	数 据 类 型	长度	说　　明
id	int		序号、主键、自动增长
goodsid	varchar	20	商品编号
quantity	int		进货数量
storenum	int		分店编号
applydate	timestamp		请求时间
processdate	timestamp		处理时间
size	varchar	5	尺码
orderstatus	varchar	10	订单状态:待处理/已发货/待发货/取消

3.3 任务三：添加记录

任务目标：

为新创建的数据表 goods 添加新记录。

技能训练：

- SQLyog 的使用；
- 数据表中记录的添加；
- 数据表中记录的删除；
- 保存数据表的修改。

下面为数据表 goods 添加新记录，如图 3-6 所示。



图 3-6　打开数据表

数据表在右侧栏中被打开，在各列中输入相应的信息就可以添加新记录了，如图 3-7 所示。

id	name	image	price	color	size
JYW12CKN1	牛仔裤	JYW12CKN1.jpg	123	水洗黑	S-XL
JYW12CKY2	牛仔裤	JYW12CKY2.jpg	158	黄	S-XL
JYW12QKB2	牛仔裤	JYW12CKB0.jpg	98	蓝	S-L
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

图 3-7　录入表中的记录

如果要删除一条记录，选中该记录前面的复选框，单击红色垃圾箱标识的 Delete selected row(s)功能键即可，如图 3-8 所示。

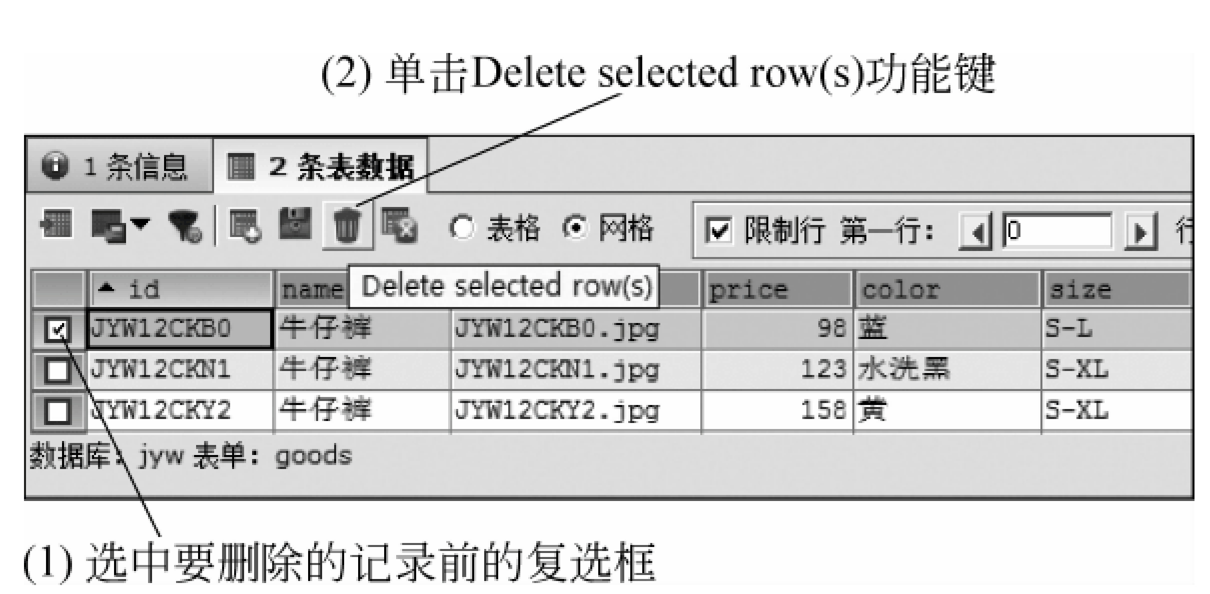


图 3-8　删除表中的记录

数据表中的数据改变后，应及时保存，单击软盘标识的 Save changes 功能键即可，如图 3-9 所示。

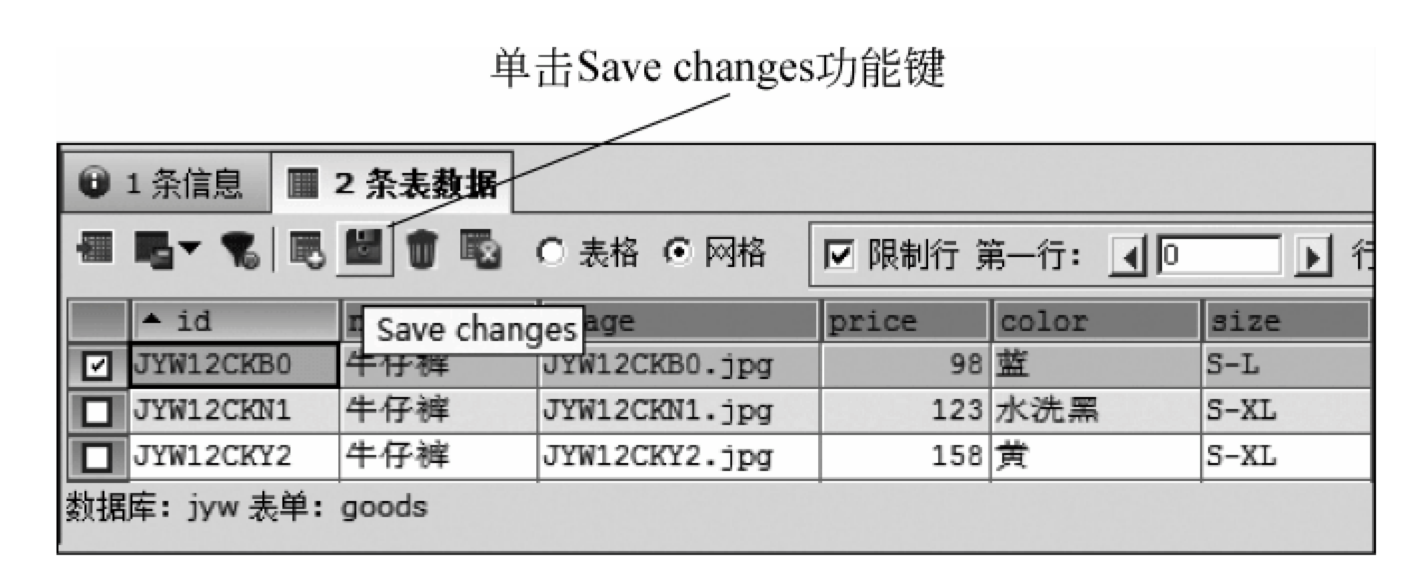


图 3-9　保存数据表所做的修改

3.4　任务四：创建视图

任务二中创建的基本信息表并不能满足所有的应用需求，有时，所需要的信息是从两个或多个基本表中得来的，这就需要用视图来表示。例如要得到销售商品的信息，其中一部分信息(如销售时间)从销售情况表中得来，另一部分从商品信息表中得来的(如商品名称)。本节讲授在 SQLyog 中如何创建视图。

任务目标：

创建视图 salesinfo，其中商品编号、名称、图片、颜色、价格来自数据表 goods，销售日期、尺码、数量、销售人员、销售数量来自数据表 sales，goods 表中的指导价格乘以 sales 表中的折扣再除以 100，即为视图中的销售价格。

技能训练：

- SQLyog 的使用；
- 视图的创建；
- SQL 脚本的编写。

图 3-10～图 3-12 以图形向导的方式介绍了如何在 SQLyog 中创建一个视图，由于编写 SQL 命令集(脚本)属于数据库课程的内容，在本书中不做详细讲解。

(1) 右击“视图”菜单项

(2) 单击级联菜单项“创建视图”

jyw 表 视图 存 函 触 事 library mysql 创建视图... 改变视图... F6 删除视图... Del 重命名视图 F2 导出视图... 打开表 F11

图 3-10　创建视图



图 3-11　输入新视图名称

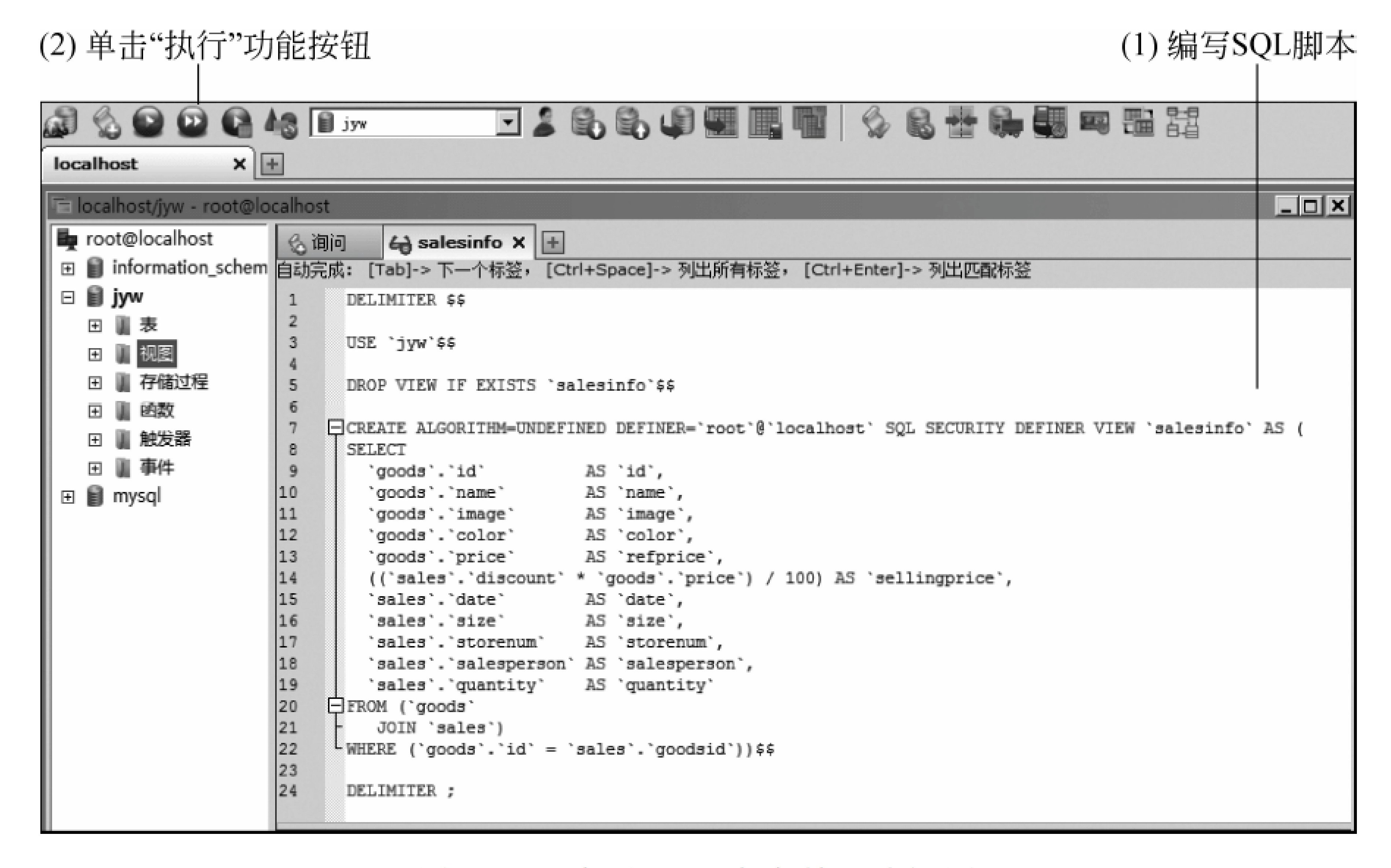


图 3-12　编写 SQL 脚本并生成视图

3.5　任务五：数据表的备份和还原

数据库要经常性地进行备份，在万一出现数据丢失的情况下还可以根据备份文件进行数据的恢复。

任务目标：

使用 SQLyog 为 jyw 数据库进行备份，并可由备份进行数据的还原。

技能训练：

- SQLyog 的使用方法；
- 数据库的备份和还原。

如图 3-13 和图 3-14 所示是 SQLyog 中进行数据库的操作步骤。

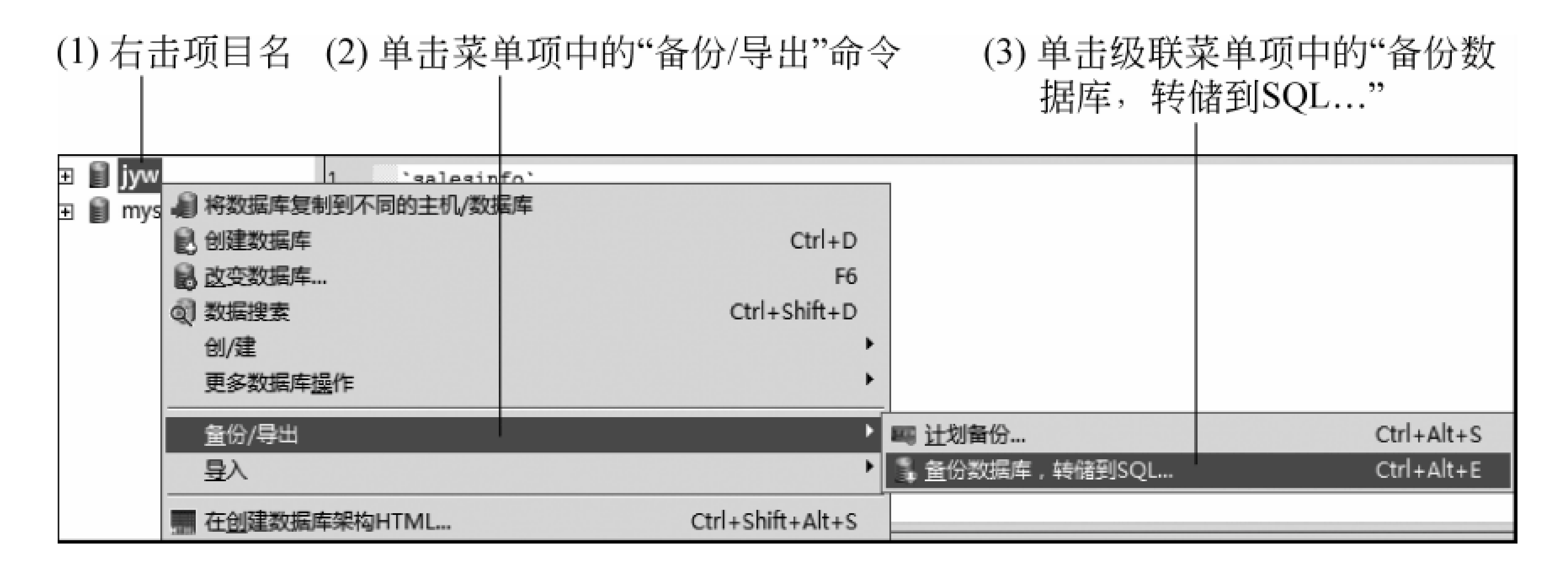


图 3-13　数据库的备份



图 3-14　导出 SQL 文件

完成以上的步骤后，数据库的备份就完成了。

可以把数据库 jyw 删除，再通过图 3-15～图 3-17 的操作进行恢复。

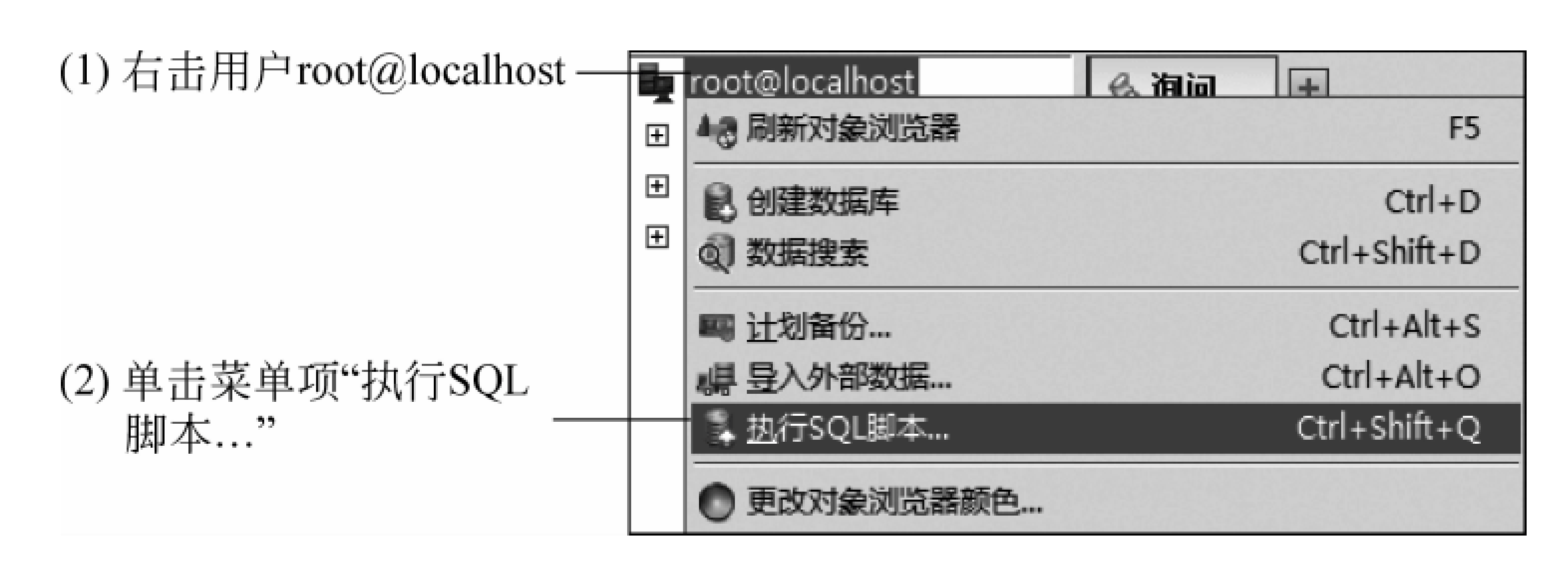


图 3-15　编写 SQL 命令生成视图

单击图 3-15 中的菜单项"执行 SQL 脚本…"后，出现如图 3-16 所示的对话框。

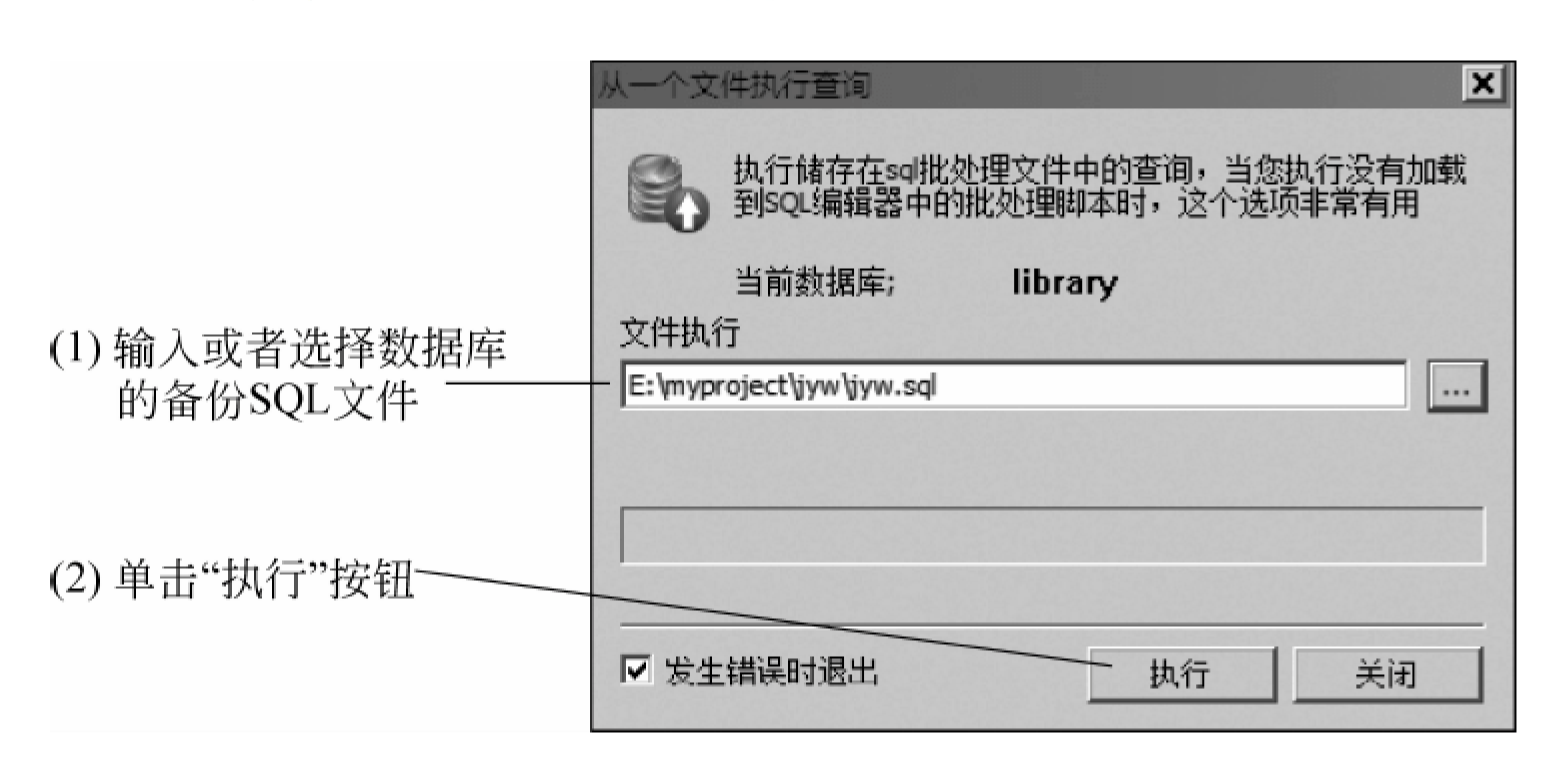


图 3-16　编写 SQL 命令以便生成视图

单击图 3-16 中的"执行"按钮后，会出现一个进度条，提示数据恢复的进度。恢复完成后，还需要进行主动刷新，恢复的数据库才会出现在左侧的对象浏览器中，如图 3-17 所示。



图 3-17　刷新对象浏览器

在刷新对象浏览器中可以看到恢复的数据库，读者可以打开数据库的数据表，查看数据是否与做备份时相同。

3.6　任务六：Tomcat 数据库连接池的配置

数据库连接是一种关键且有限的资源，这一点在多用户的 Web 应用中体现得尤为突出。对数据库连接的管理能显著影响到整个应用程序的可扩展性和健壮性，影响到 Web 应用的性能指标。数据库连接池正是针对这个问题提出来的。数据库连接池负责分配、管理和释放数据库连接，它允许应用程序重复使用一个现有的数据库连接，而不是重新建立一个；通过释放空闲时间超过最大空闲时间的数据库连接来避免因为没有释放数据库连接而引起数据库的连接问题。这项技术能明显提高数据库操作的性能。

为 Tomcat 配置 MySQL 数据库连接池，需要在 Tomcat 的 context.xml 文件中的 <context></context>之间添加 MySQL 连接池配置，操作如下。



```
<Resource   name="jdbc/mysql"
            auth="Container"
            type="javax.sql.DataSource"
            maxActive="50"
            maxIdle="10"
            maxWait="5000"
            username="root"
            password="password"
            driverClassName="org.gjt.mm.mysql.Driver"
            url="jdbc:mysql://localhost/jyw" />
```


与 MySQL 的连接还需要 MySQL 驱动的支持，请读者自行下载 MySQL 的 jdbc 驱动 jar 包 mysql-connector-java-5.0.2-beta-bin.jar，并将其复制到 tomcat 下的 lib 目录下。

完成本章任务六后，数据库的连接池配置就完成了。需要注意的是，任务六中的配置是针对 Tomcat 7.0，Tomcat 6.0 及以下版本的配置方法会有不同。作为开源 Web 服务

器，Tomcat 得到了广泛的应用，针对 Tomcat 的技术讨论在互联网上也很活跃，请使用 Tomcat 6.0 及以下版本的读者自行查找资料并完成连接池的配置。

课 后 练 习

1. 请读者下载 MySQL 5.0 及以上版本、SQLyog 9.0 及以上版本并完成安装。

2. 请按照本章任务二中的表 3-1～表 3-5 完成各数据表的创建，并进行信息的初始化录入。

3. 在练习 2 的基础上，从基本表 goods 和 orders 中导出订单详细信息视图 ordersinfo。

要求：商品的编号、名称、图片、颜色、价格来自商品信息数据表 goods，序号、尺码、数量、请求日期、处理日期、订单状态、分店号来自订单数据表 orders，其条件为 goods 表中的 id 与 orders 表中的 goodsid 相等。

提示：参考本章任务四，完成视图 ordersinfo 的创建。

4. 在练习 2 的基础上，导出详细库存信息视图 stockinfo。

要求：以商品信息表 goods、库存信息表 stock、分店信息表 storeinfo 为基本表，导出图 3-18 中的视图。

	id	name	image	price	color	size	storename	stock
□	JYW12CKN1	牛仔裤	JYW12CKN1.jpg	123	水洗黑	M	佳衣屋总店华强北店	38
□	JYW12CKY2	牛仔裤	JYW12CKY2.jpg	158	黄	M	佳衣屋总店华强北店	30
□	JYW12CKN1	牛仔裤	JYW12CKN1.jpg	123	水洗黑	M	佳衣屋海岸城店	20
*	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

图 3-18　视图

提示：参考本章任务四，完成视图 stockinfo 的创建。

5. 根据第 1 章中的需求分析，完成“佳衣屋”项目数据库的设计。

第 4 章　知 识 准 备

本章学习要点：

- 熟练掌握表单及其常用元素的用法；
- 熟练掌握 JSP 指令标签的用法；
- 熟练掌握 JSP 动作标签的用法；
- 熟练掌握内置对象 request 的常用方法的用法；
- 熟练掌握内置对象 response 的常用方法的用法；
- 熟练掌握内置对象 session 的常用方法的用法；
- 熟练掌握内置对象 application 的常用方法的用法；
- 熟练掌握内置对象 out 的常用方法的用法；
- 了解 JDBC 的结构和原理；
- 熟练掌握 JDBC 操作数据库的步骤。

本章内容涉及 HTML 表单及其元素、JSP 基本概念、JSP 内置对象和 JDBC 数据库操作等，是后面章节中的“佳衣屋”项目开发过程中所使用的技术的知识基础，已经熟悉本章内容的读者可以直接进入第 5 章的学习。或者在后续章节的项目开发中遇到不明白的概念时，再回到本章寻找参考内容。

4.1　HTML 表单及其元素

HTML 表单和元素用于为 Web 应用搜集不同类型的用户输入，表单元素必须包含在表单中才能有效使用。

4.1.1　表单

表单是一个包含表单元素的区域，表单使用表单标签（<form></form>）定义。表单可选的属性见表 4-1。

（1）action 属性。用于指定此表单数据提交的目标 URL，它通常是一个相对路径。如果没有设置这个属性或者属性值为空，表单数据将提交给此表单自身的 URL。

表 4-1　表单属性

属　性	值	描　　述
action	URL	确定当提交表单时,向何处发送表单数据
accept	MIME_type	确定通过文件上传来提交的文件的类型
accept-charset	charset	服务器处理表单数据所接受的字符集
enctype	MIME_type	确定表单数据在发送到服务器之前应该如何编码
method	get|post	确定如何发送表单数据
name	name	确定表单的名称
target	_blank|_parent|_self|_top|framename	确定在何处打开 action URL

(2) method 属性用于指定提交表单数据的方式,常用的有 GET 和 POST 两种方式。如果没有设置此属性或者此属性值为空,则使用 GET 方式来提交数据。GET 和 POST 提交方式的主要不同点在于如下三点。

① 因为 GET 数据是 URL 的一部分,所以它会将表单数据附在 URL 后面传送。也就是说,在浏览器的地址栏将会显示表单中的数据,并且在通常情况下,浏览器会将这个附加数据后的 URL 保存起来,可以通过浏览器的"历史"来得到它。所以,这种方式不适合于发送需要保密的数据的表单,比如密码等。而 POST 不是 URL 的一部分,所以它不会将表单数据附在 URL 后面,所以这种方式不会发生上面的问题。从这种意义上讲,POST 方式比 GET 方式更安全。

② 因为浏览器通常会限制 URL 的长度,所以使用 GET 这种方式无法传送大量的数据。而 POST 方式不会有这种问题,所以 POST 方式传输的数据量大。

③ 由于 GET 传输的数据量小,所以有些数据类型,比如文件的传输只能用 POST 方式。从这一点上讲,POST 方式可传输的数据类型更多。

所以,如果没有特殊的需要,应使用 POST 方式来传送表单数据。

(3) name 属性。用于给这个 FORM 指定一个名字,可以用字母和数字组合的方式来给 FORM 命名,但不要用数字开头。

(4) enctype 属性。用于定义数据在发送前需要完成的编码方式,如果没有设置这个属性,那么会使用默认的值 application/x-www-form-urlencoded,它使用的编码方式是 UTF-8。

(5) accept 属性。用于指定处理表单数据的 ASP、JSP、Servlet 或者其他程序接受的 MIME(Multipurpose Internet Mail Extension protocol,多用途网际邮件扩充协议)数据类型,如果 FORM 中有文件组件(FILE),还可以使用它来限制上载文件的类型。

(6) accept-charset 属性。用于指定处理表单数据的 ASP、JSP、Servlet 或者其他的程序接受的字符编码。

4.1.2　表单元素

表单元素在 Web 应用中用来给访问者填写信息,从而能采集客户端信息,使 Web 应

用具有交互的功能。常用的表单元素有以下几种。

1. 文本框

文本框定义的基本格式如下：

```
<INPUT TYPE="TEXT" NAME="" VALUE="" SIZE="" MAXLENGTH="">
```

文本框用标记 INPUT 定义，并且需要将它的 TYPE 属性值定义为 TEXT，这也是 INPUT 标记的默认类型；NAME 属性用于给文本框指定一个名字，这个属性是必需的；VALUE 属性可以用于指定文本框的默认值；SIZE 属性用于定义文本框的大小，默认是 20；而 MAXLENGTH 是用于限制文本框的输入数据长度的属性。

2. 密码框

密码框和文本框类似，和文本框唯一的区别是，需要将 INPUT 标记的 TYPE 属性设置为 PASSWORD，其他的属性设置和文本框的设置一样。另外，在密码框中输入数据的时候，密码框中不会明文显示输入的数据，而是用“*”或其他的掩盖字符来表示，但这并不影响将输入的数据进行发送。

3. 文本域

文本域使用<TEXTAREA>标记来定义，它的基本格式如下：

```
<TEXTAREA NAME="" ROWS="" COLS=""></TEXTAREA>
```

与前面的两个文本元素不同，<TEXTAREA>标记必须成对出现，它有三个属性必须定义：NAME 用于设置文本域的名字，ROWS 用于设置文本域的行数，而 COLS 用于设置文本域的列数，如果文本域有默认值，则将默认值放在<TEXTAREA>和</TEXTAREA>之间。

4. 列表菜单

列表菜单向用户提供一系列的选项。它可以分为单选列表和多选列表两种。单选列表可以让用户选择一个选项，它也是下拉列表的默认设置；多选列表可以让用户选择多个选项。

列表菜单的基本格式如下：

```
<SELECT NAME="" SIZE="">
        <OPTION VALUE="#">#</OPTION>
</SELECT>
```

列表菜单使用 SELECT 标记来定义，需要使用 NAME 属性来给它指定一个名字。SIZE 属性用于指定下拉列表在浏览器中显示的行数，如果不指定这个属性，那么在浏览器中只有一行可见。下拉列表的各个选项可以使用<OPTION>标记来定义，使用 VALUE 属性来给选项指定值，这个值不会显示在浏览器中。如果需要指定一个默认的

选项，可以在该选项中加上一个 SELECTED 属性。而在<OPTION></OPTION>之间，可以定义显示到浏览器中的内容。

如果需要定义多选列表，可以通过指定<SELECT>标记的一个属性 MULTIPLE 来完成，并且同时将 SIZE 属性值设置为 2 或者 2 以上。

示例 4-1 是一个单选列表和多选列表的表单元素用法的例子。

示例 4-1　单选列表和多选列表

```
<HTML>
    <BODY>
        <FORM NAME="FN" ACTION="">
            <SELECT NAME="Favorite" SIZE="4" MULTIPLE>
                <OPTION VALUE="Basketball">篮球</OPTION>
                <OPTION VALUE="Volleyball">排球</OPTION>
                <OPTION VALUE="Table Tennis">乒乓球</OPTION>
                <OPTION VALUE="Tennis">网球</OPTION>
            </SELECT>
            <BR><BR>
            <SELECT NAME="Gender">
                <OPTION VALUE="Man" SELECTED>男</OPTION>
                <OPTION VALUE="Woman">女</OPTION>
            </SELECT>
        </FORM>
    </BODY>
</HTML>
```

图 4-1 是这两个列表的显示效果，第一个为多选列表，第二个为单选列表。在第二个单选列表中，指定了默认选项为第一个选项。



图 4-1　示例 4-1 的显示效果

5. 单选框

单选框提供给用户多选一的组件，它的基本格式如下：

```
<INPUT TYPE="RADIO" NAME="#" VALUE="#">#
```

单选框也是使用<INPUT>标记来定义，但是需要将它的 TYPE 属性设置为 RADIO，需要给它指定一个名字。一个<INPUT>只能定义一个单选框选项，因此对于一组的选项，必须针对每个选项定义并且需要给它们提供一样的名字，这样，这些选项就能组成一个“组”。在这些选项中，每次最多只能有一个选项被选中。如果需要指定默认的选项，可以给该选项指定 CHECKED 属性，如：

```
<INPUT TYPE="RADIO" NAME="性别" VALUE="男">男
<INPUT TYPE="RADIO" NAME="性别" VALUE="女" CHECKED>女
```

这样，对于“性别”单选框，它有两个选项，默认选项是“女”。

6. 多选框

多选框提供给用户一个选择多个选项的组件，它的基本格式如下：

```
<INPUT TYPE="CHECKBOX" NAME="#" VALUE="#">#
```

通常情况下，将一组同样性质的多选框指定用一样的名字。如果需要指定某些选项为默认选项，可以给这个选项指定 CHECKED 属性。被选择中的各个选项值会组成一个字符串并被发送到服务器端，各个选项值之间用逗号隔开。

下面是一个多选框和单选框的表单元素用法的例子。

示例 4-2 单选框和多选框

```
<HTML>
  <BODY>
    <FORM NAME="FN" ACTION="">
      选择性别：
      <INPUT TYPE="RADIO" NAME="性别" VALUE="男">男
      <INPUT TYPE="RADIO" NAME="性别" VALUE="女" CHECKED>女
      <BR><BR>
      选择你的兴趣爱好：
      <INPUT TYPE="CHECKBOX" NAME="兴趣" VALUE="上网" CHECKED>上网
      <INPUT TYPE="CHECKBOX" NAME="兴趣" VALUE="踢球">踢球
      <INPUT TYPE="CHECKBOX" NAME="兴趣" VALUE="篮球">篮球
      <INPUT TYPE="CHECKBOX" NAME="兴趣" VALUE="健身" CHECKED>健身
    </FORM>
  </BODY>
</HTML>
```

图 4-2 是它们的显示效果。



图 4-2 示例 4-2 的显示效果

7. 按钮

在 HTML 中，有三种类型的按钮：Submit、Reset 和 Button。使用 Submit 按钮可以将表单提交到 FORM 标记的 ACTION 所指定的 URL 中；Reset 按钮可以将表单的内容恢复到原始的状态；而 Button 类型的按钮通常情况下需要和 JavaScript 结合起来使用才有意义。

- Submit

Submit 按钮定义的基本格式如下：

```
<INPUT TYPE="SUBMIT" NAME="#" VALUE="#">
```

可以看出，Submit 按钮的定义也是使用＜INPUT＞标记，只是需要将 TYPE 属性值指定为 SUBMIT 就可以了，另外需要给它指定一个名字。VALUE 属性将显示在 Submit 按钮上，但它不是必需的，如果没有指定它，那么在浏览器中就会自己加上一个值，在不同的浏览器中可能会有所不同，如在英文的 IE 中通常会是 Submit Query，而在中文 IE 中是“提交查询内容”等。

- Reset

Reset 按钮用于将表单中的各个组件的值恢复到最初设置的初值，它的基本格式如下：

```
<INPUT TYPE="RESET" NAME="#" VALUE="#">
```

RESET 按钮的定义也使用＜INPUT＞标记，只是需要将 TYPE 属性值指定为 RESET 就可以了，另外需要给它指定一个名字。VALUE 属性将会显示在 Reset 按钮上，它可以不指定，而是让浏览器自己给它赋值，不同的浏览器中会有一些差别，如在英文 IE 中显示为 Reset，而在中文 IE 中显示为“重置”等。

- Button

Button 通常需要和 JavaScript 结合起来才起作用，因为它本身不能完成任何的功能，它的格式如下：

```
<INPUT TYPE="BUTTON" NAME="#" VALUE="#">
```

与 Submit 和 Reset 按钮一样，Button 使用＜INPUT＞来定义。在 Button 定义中，需要给 VALUE 指定一个值；否则，在浏览器中的按钮组件上不会有任何的文字指示。

在 FORM 中，通常至少需要定义一个 Submit 按钮，这样表单才可以提交到 ACTION 指定的 URL 中。当然，使用 Button 按钮和 JavaScript 的组合也能达到一样的目的。

课堂练习 4-1　制作用户登录页

要求登录页中含有：

- 一个文本框，用于输入用户名；
- 一个密码框，用于输入密码；
- 含有一个“提交”按钮，按钮上显示“登录”；
- 含有一个指向 register.jsp 的按钮，按钮上显示“注册”；
- 当单击“提交”按钮时，表单数据提交给 loginCheck.jsp 处理。

示例 4-3　登录页 login.jsp

```
<center>
<FORM name="form1" method="post" action=" "></P>
    <P>用户名<INPUT type="text" name="username" '></P>
    <P>密码  <INPUT type="password" name="password"></P>
    <P><INPUT type="submit" name="submit" value="提交" />
        <INPUT type="button" name="button" value="注册"
                  onclick="document.location.href='register.jsp'" />
```

```
    </P>
</FORM>
```

将示例 4-3 中的 login.jsp 复制到 Tomcat 的根目录下，启动 Tomcat 服务器，在浏览器的 URL 中输入 http://localhost:8080/login.jsp，会看到如图 4-3 所示的执行效果。



图 4-3　login.jsp 的显示效果

课堂练习 4-2　制作用户注册页

要求注册页中含有如下内容：

- 一个文本框，用于输入用户名；
- 一个密码框，用于输入密码；
- 一组单选按钮，用于选择性别；
- 一个文本区域，用于输入用户的自我介绍；
- 一组单选列表菜单，用于选择用户所在的学院；
- 一组复选框，用于用户选择爱好；
- 一个“提交”按钮，按钮上显示为“注册”；
- 当单击“提交”按钮时，表单数据提交给 dealRegister.jsp 处理。

示例 4-4　登录页 register.jsp

```
<form id="form1" name="form1" method="post" action="dealRegister.jsp">
<p><label>username
  <input type="text" name="username" />
    </label></p>
  <p><label>password
  <input type="password" name="password" />
    </label></p>
  <p><label>gender male
    <input type="radio" name="gender" value="male" />
    </label>
    <label>female
    <input type="radio" name="gender" value="female" />
    </label></p>
  <p>
    <label>introduction
    <textarea name="introduction" rows="6"></textarea>
```

```
    </label></p>
  <p>
    <label>school
    <select name="comefrom">
      <option>Computer Sciences</option>
      <option>Mechatronic Engineering</option>
      <option>Physical Science</option>
    </select>
    </label></p>
  <p>
    <label>hobby
     music  <input type="checkbox" name="checkbox" value="music" />
  </label>
  <label>
  sports  <input type="checkbox" name="checkbox" value="sports" />
  </label>
  <label>
  dancing  <input type="checkbox" name="checkbox" value="dancing" />
  </label></p>
<p>
  <label>
  <input type="submit" name="Submit" value="register" />
  </label></p>
</form>
```

示例 4-4 的显示效果如图 4-4 所示。

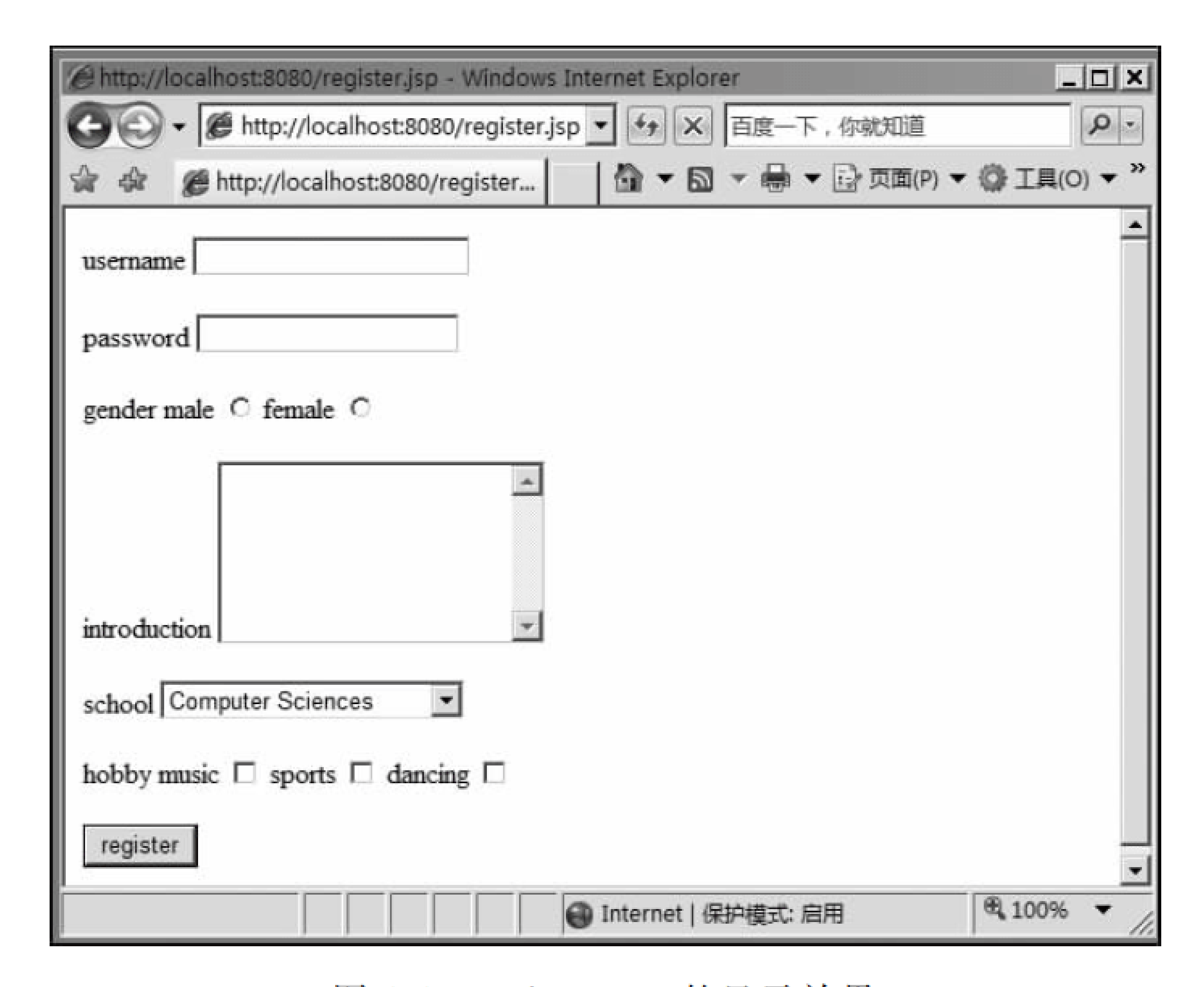


图 4-4　register.jsp 的显示效果

8. 隐藏域

隐藏域可以定义在 FORM 中，用来传递不需用户输入的值。与其他的 FORM 组件

不同，它不会显示在浏览器中，用户不能去修改它的值。它的定义基本格式如下：

```
<INPUT TYPE="HIDDEN" NAME="#" VALUE="">
```

与文本框等组件一样，隐藏域的定义也是使用<INPUT>标记来完成的，需要将它的 TYPE 属性值设置为 HIDDEN，并且需要给它指定一个名字。另外因为隐藏域不能接收用户的输入，所以通常需要给它指定一个 VALUE 值。

9. 文件上载组件

有时候 Web 应用需要将客户端的文件上载到服务器端，这时候就需要使用文件上载组件来接收需要上载的文件的路径，它的基本格式如下：

```
<INPUT TYPE="FILE" NAME="#" MAXLENGTH="#" SIZE="#">
```

文件上载组件也使用<INPUT>标记来定义，并且将它的 TYPE 属性值设置为 FILE，同时需要给它的 NAME 选项指定一个值。MAXLENGTH 和 SIZE 属性的含义和文本框的含义一样。文件上载组件在浏览器上的表现形式为一个“文本框”和一个“按钮”，这个按钮会在不同的浏览器上面显示类似“浏览...”的内容。单击这个按钮，将打开一个文件选择对话框，让用户选择一个文件。

另外，如果要让文件能够顺利上载，需要将对应的 FORM 的 ENCTYPE 属性设置成 multipart/form-data，并且一定需要将 METHOD 属性设置为 post。

在示例 4-4 的“提交”按钮前添加如下的文件上载组件。

```
<p><label>picture <input type="file" name="picture"></label></p>
```

图 4-5 为添加了文件上载组件后的注册页在 IE 上的显示效果。

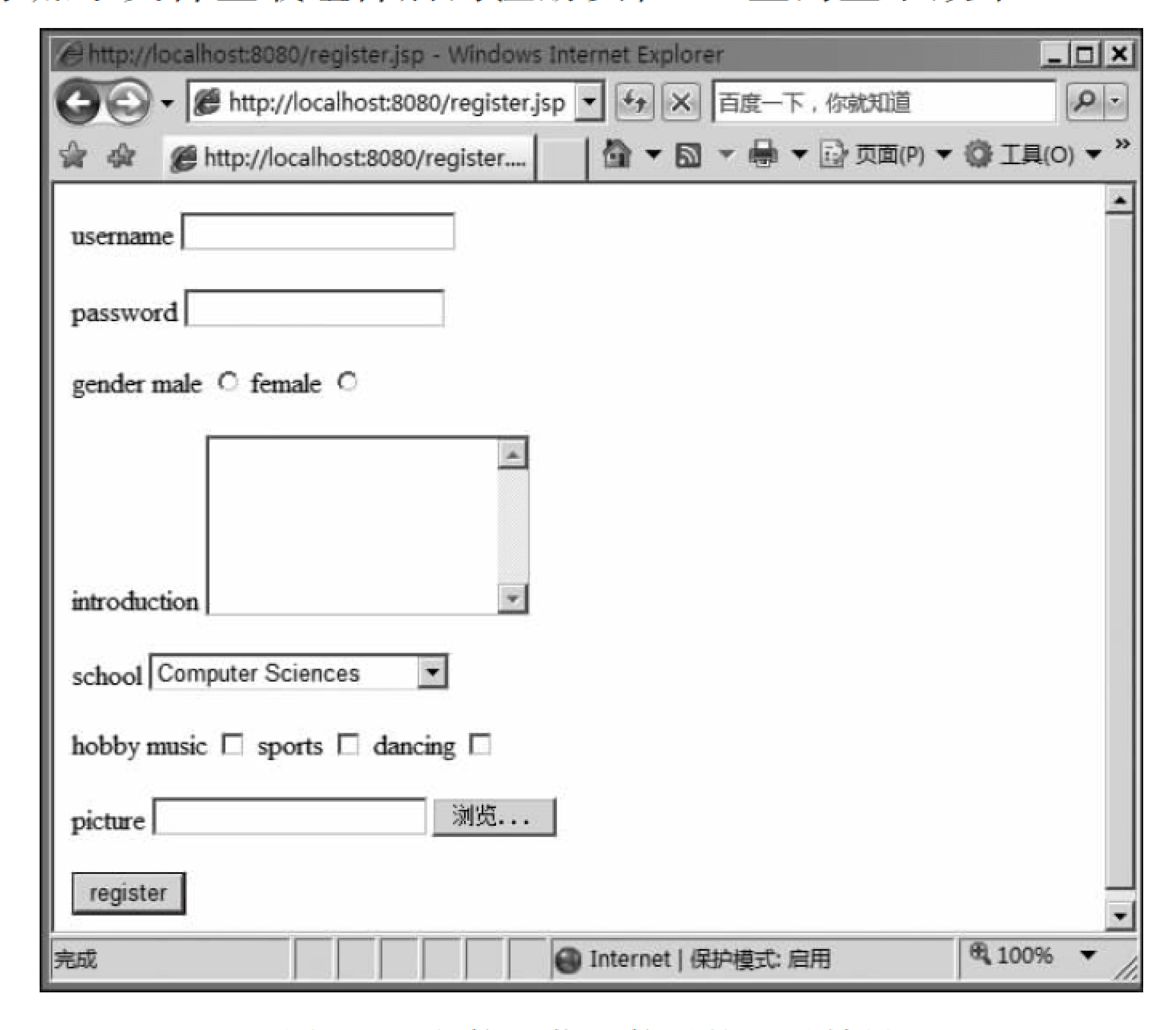


图 4-5　文件上载组件后的显示效果

4.2　JSP 语 法

JSP(Java Server Pages)是由 Sun Microsystems 公司倡导、许多公司参与并共同创建的一种使软件开发者可以响应客户端请求，而动态生成 HTML、XML 或其他格式文档的 Web 网页的技术标准。JSP 技术是以 Java 语言作为脚本语言的，JSP 网页为整个服务器端的 Java 库单元提供了一个接口来服务于 HTTP 的应用程序。JSP 网页(*.jsp)就是在传统的 HTML 文件(*.htm，*.html)中加入 Java 程序片段和标记而构成的。

4.2.1　JSP 页面的组成

要想在 JSP 页面中实现丰富的动态功能，需要 JSP 各种组成成分的帮助，常用的组成成分及功能说明见表 4-2。

表 4-2　主要 JSP 页面组成

元　素	描　述
注释	在客户端建立一个可见或不可见的注释
声明	在脚本语言里声明变量和方法
表达式	在脚本语言里加入表达式
程序片	在脚本语言里加入脚本程序段
包含指示	加入一个用来说明 JSP 基本要素的文件
页面指示	定义整个 JSP 页面的属性
标签指示	在 JSP 页面里定义标签库和定制标签的前缀

4.2.2　JSP 注释

JSP 注释可以大致分为两种：一种是可以发送到客户端的，一种是不会发送给客户端而仅提供给 JSP 程序员阅读的，这两种注释在语法上是不同的。

1. 发送到客户端的 JSP 注释

发送到客户端的 JSP 注释的基本语法格式如下：

```
<!--注释 [<%=表达式%>]-->
```

这种方式其实是 HTML 注释方式，只不过可以在里面加上 JSP 表达式，动态生成注释内容。

示例 4-5　发送到客户端的注释 jspComments1.jsp

```
<%@page contentType="text/html;charset=gb18030"%>
```

```
<HTML>
<HEAD>
    <TITLE>发送到客户端的注释</TITLE>
</HEAD>
<BODY>
    <!--这个注释会发送到客户端<%=new java.util.Date()%>-->
</BODY>
```

其中“＜！——”和“——＞”部分内容不会在浏览器中显示，如图 4-6 所示。



图 4-6　示例 4-5 的执行结果

使用浏览器查看源文件的功能，可以看到如图 4-7 的结果，表明“＜！——”和“——＞”部分的内容已经发送到了客户端。

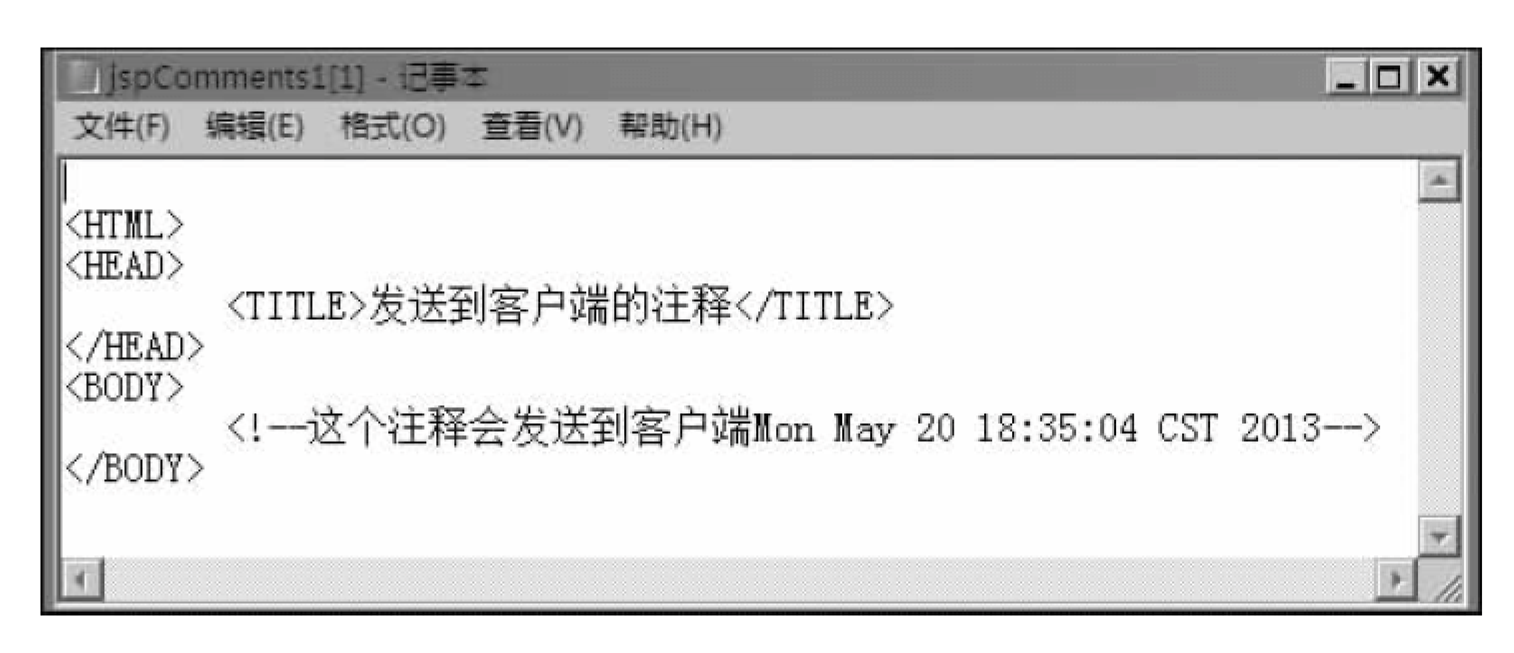


图 4-7　在浏览器中查看页面源文件的结果(一)

2. 不发送到客户端的注释

不发送到客户端的注释的基本格式如下：

```
<%--注释内容--%>
```

在“＜％——”和“——％＞”部分的内容只是为方便程序员或其他相关人员阅读代码而添加的注释，不会被编译，不会被执行，也不会被发送到客户端。示例 4-6 中的注释语句就是这种注释。

示例 4-6　jspComments2.jsp

```
<%@page contentType="text/html;charset=gb18030"%>
<HTML>
<HEAD>
        <TITLE>不发送到客户端的注释</TITLE>
</HEAD>
```

```
<BODY>
        <%--这个注释不会发送到客户端<%=new java.util.Date ()%>--%>
</BODY>
```

示例 4-6 运行后，在客户端浏览器中查看它的源文件，内容如图 4-8 所示。



图 4-8　在浏览器中查看页面源文件的结果(二)

可以看出，这个注释没有发送到客户端。

因为在 JSP 程序中，有一部分是 Java 代码，因此，除了上面两种 JSP 的注释方式以外，在 Java 代码中，还可以像普通的 Java 类中一样使用 Java 的注释语句，单行注释用"//"，而多行注释使用"/ * "和" * /"包含起来。推荐使用"//"的注释方式，因为在某些集成开发环境如 MyEclipse 中，这种注释方式会使用绿色等颜色特别显示，使开发人员容易将注释与代码的其他部分区分开来。

4.2.3　JSP 程序片

如果业务逻辑比较复杂，JSP 程序中可能含有大段的 Java 代码，这时就要使用 Java 程序片。Java 程序片就是用"＜％"和"％＞"包含起来的 Java 代码段。格式如下：

```
<%
Java 代码
%>
```

在 JSP 中，可以在任何地方插入任何数量的 Java 程序片。但要注意的是，Java 程序片之间不能嵌套。

课堂练习 4-3　Java 程序片的使用

要求：将 1～10 累加的结果显示在 IE 浏览器中。

提示：JSP 中的 Java 程序片包含在＜％　％＞标签中。

示例 4-7　Java 程序片实例 doSum.jsp

```
<HTML>
<HEAD>
        <TITLE>1-10 的累加</TITLE>
```

```
</HEAD>
<BODY>
        <%
        int sum=0;
        for(int i=1;i<=10;i++)
        {
            sum+=i;
        }
        out.print(sum);
        %>
</BODY>
</HTML>
```

Java程序片

示例中的 out. print(sum)的作用是向浏览器中输出 sum 的值,out 是一个内置对象。关于 jsp 中的内置对象,将在后续小节中学习。

图 4-9 是这个程序的运行结果。



图 4-9 doSum. jsp 的运行结果

4.2.4 JSP 声明

在 JSP 中,方法的声明以及全局变量的声明应该放在 JSP 声明部分,它的基本格式如下:

```
<%!声明 %>
```

JSP 声明中的变量是全局变量,所以它可以在任何地方被使用。

下面来看一个 JSP 声明的例子。

示例 4-8 JSP 声明效果演示 declaration. jsp

```
<HTML>
<HEAD>
<TITLE>declaration</TITLE>
</HEAD>
<BODY>
    <%!
    private String str="This is globalvariable";
    private int sum(int n)
    {
```

全局声明

```
        int result=0;
        for(int i=1;i<=n;i++)
        {
          result+=i;
        }
        return result;
    }
    %>
    <%
            int s=sum(10);
            out.print("The result is :"+s);
    %>
    <hr>
    <%=str%>
</BODY>
```

这个例子在 JSP 声明中定义了一个全局变量 str 和一个方法 sum ()，这个方法完成用户从 1 到 n 的累加。定义方法后，可以在 java 程序片中直接调用这个方法，或者可以在另一个方法声明中调用它。

图 4-10 是这个程序的运行结果。



图 4-10　declaration. jsp 运行结果

4.2.5　JSP 表达式

JSP 表达式的基本语法如下：

```
<%=expression%>
```

其中，expression 是一个可以转换成字符串的表达式。JSP 表达式的作用是将 expression 的运算结果输出到客户端。使用表达式有几个地方需要注意：

(1) JSP 表达式只能单独出现，也就是说，不能在一对“<%”和“%>”之间出现多条语句；

(2) JSP 表达式和普通的 Java 语句不同，不能用“;”结束；

(3) JSP 表达式的“<%”、“%>”和 expression 只能在同一行上；

(4) 表达式 expression 必须能够转换成字符串形式；

(5) 不在表达式中定义变量和方法；

(6) 不能在表达式中使用返回值为 void 的方法。

使用表达式的效果和在 Java 程序片中使用 out.println()的效果是一样的，它们都会向客户端输出内容。事实上，在 JSP 中，也可以使用一个内置的 out 对象来实现这个功能。关于 out 的使用，请读者参考后续章节的内容。

课堂练习 4-4　动态输出表格内容

要求：在一个三行两列的表格中输出 5、10、15 的累加值。

5	15
10	55
15	120

提示：

- 全局变量声明和表达式的使用，方便实现动态内容按设计者的要求出现在网页的不同部位。
- Html 标签和 Java 程序片可交替出现，但 Html 标签不能被包含在 Java 程序片内。

示例 4-9　JSP 表达式效果演示 expression.jsp

```
<table border="1">
<%
int sum=0;
for(int j=1;j<=3;j++){
    for(int i=1;i<=j * 5;i++)
    {
          sum+=i;
    }
    %>
    <tr>
        <td><%=j * 5%></td>         JSP表达式的使用
        <td><%=sum%></td>
    </tr>
    <%
    sum=0;
}
%>
</table>
```

示例 4-9 的显示效果如图 4-11 所示。



图 4-11　expression.jsp 的显示效果

4.3　JSP 指令标签

JSP 指令标签主要用来提供整个 JSP 网页相关的信息，并且用来设定 JSP 页面的相关属性。JSP 指令的语法格式如下：

```
<%@directive {attr=value} %>
```

directive 为标签的名称。JSP 有三个指令标签：page、include 和 taglib，本书只介绍 include 和 page 指令。taglib 指令用于指定需要用到的标记库。attr 部分为指令标签的属性。

4.3.1　page 指令

page 指令允许程序员导入需要的类、指明 JSP 输出内容类型、控制 session 等。其语法格式如下：

```
<%@page {attr=value} %>
```

下面介绍 page 指令的主要属性。

1. contentType 属性

contentType 属性用于指明 JSP 输出的内容的 MIME(Multipurpose Internet Mail Extensions)类型。在默认情况下，JSP 的 MIME 类型是“text/html;charset＝ISO-8859-1”，中文网页常见的 MIME 类型有“text/html; charset＝GB2312”、“text/html;charset＝GBK”或者“text/html;charset＝GB18030”。

课堂练习 4-5　解决 JSP 页面的中文乱码问题

要求：图 4-9 左上角的网页标题中出现了中文乱码，将这一问题更正，使中文正常显示。

提示：在 page 指令中设置 contentType 来指定本页采用中文简体编码集。

将下面的 page 指令添加到 doSum.jsp 的第一行。

```
<%@page  contentType="text/html;charset=gb18030" %>
```

修改后的 doSum.jsp 的显示效果如图 4-12 所示。与图 4-9 相比，可以看到已经更正了页面标题的乱码问题。

2. pageEncoding 属性

通过 contentType 可以设置 JSP 输出结果的 MIME 类型，如果只想改变文件的编码(使用默认的 text/html 内容类型)，那么只需要使用 pageEncoding 就可以了，代码如下：



图 4-12　更正了中文乱码后的显示效果

```
<%@page pageEncoding="gb18030" %>
```

3. import 属性

如果编写过 Java 程序的读者都应该知道，在 Java 程序中，通常需要用到 JDK 中的其他类，这个时候，就需要将这个类导入（import），那么，既然在 JSP 中也需要编写 Java 代码，应该如何导入其他的类呢？

通过 page 指令的 import 属性，可以方便地将需要用到的类引入进来，它的用法如下：

```
<%@page import="package.Class"%>
```

或者

```
<%@page import="package.*"%>
```

如果想要引入多个包中的类，可以使用下面的方法：

```
<%@page import="package1.Class"%>
<%@page import="package2.Class"%>
```

或者

```
<%@page import="package.Class1,package2.Class2"%>
```

提示：import 是 page 指令中唯一一个可以在同一个 JSP 中多次出现的属性。

4. session 属性

session 属性用于控制页面是否需要使用 session（会话），这个属性值可以为 true 或者 false。这个属性的默认值为 true，表示这个页面需要使用 session；如果属性值为 false，则表示这个页面不能使用 session。下面是 page 指令中的 session 属性设置的例子。

```
<%@page session="false" %>
```

使用了上面的 page 指令的 JSP 页面不能访问 session，或者为用户创建新的 session。关于 JSP 中的 session，将会在后面的小节中继续学习。

5. errorPage 和 isErrorPage 属性

errorPage 用于指明在这个 JSP 页面中出现未被捕获的异常时，跳转到哪个页面来处理。通常，跳转的页面需要使用 isErrorPage 来指明可以用于其他页面的错误处理。

errorPage 的用法如下：

```
<%@page errorPage="errorHandle.jsp" %>
```

通过上面的 page 指令，可以指明当这个 JSP 中出现错误的时候，会跳转到 errorHandle.jsp 来处理。在 errorHandle.jsp 中，通常需要使用 isErrorPage 属性来说明它可以用于其他页面的错误处理，代码如下：

```
<%@page isErrorPage="true" %>
```

在错误处理页面中可以使用内置的 exception 对象来获得相关的异常信息，例如使用它的 printStackTrace()方法来打印异常堆栈，如果在错误处理页面中使用了 exception 内置对象，那么，必须在这个 jsp 页面中使用 page 指令的 isErrorPage 属性，并且将它的值设置成 true，否则将无法使用 exception 这个内置对象。关于 exception 内置对象的更多知识，将在后续内容中介绍。

6. buffer 属性

buffer 属性用于指定在使用 out 内置对象向客户端输出内容的时候使用的缓冲区的大小，它的默认值是 8KB。

7. info 属性

info 属性可以用于指定对这个 JSP 页面的一些说明信息，可以通过 Servlet 的 getServletInfo()方法获得。它的用法如下：

```
<%@page info="Jsp Message"%>
```

8. isThreadSafe 属性

isThreadSafe 属性用于控制 JSP 页面是否允许并行访问，它可取两个值之一：true 或者 false，默认是 true，表示允许并行访问。在 JSP 2.0 中，这个属性不推荐使用，或者说，不推荐将它设置成 false。对于需要控制并发访问的地方，应该在代码中实现，不能依赖于将 isThreadSafe 设置成 false。

9. isELIgnored 属性

该属性是 JSP 2.0 中新引入的一个属性，它所要控制的是，忽略还是处理 JSP 2.0 中的 EL(Expression Language)。如果将这个值设置成 true，那么，将忽略 EL；如果将它设置成 false，将对 EL 进行正常的运算。需要注意的是，在只支持 JSP 1.2 的服务器中，不能使用这个属性。

10. language 属性

language 属性用于指定 JSP 所使用的脚本语言，在目前的情况下，只有一种选择，那就是 Java。

11. extends 属性

使用这个属性可以指定 JSP 所转换成的 Servlet 的父类。

4.3.2 include 指令

include 指令用于将多个 JSP 页面(或者 HTML、XML 文件)组合起来，从而成为一个“完整”的 JSP 页面。它的基本语法格式如下：

```
<%@include file="jspFile"%>
```

这个指令有一个必要的属性 file，它用于指明包含哪个文件。在一个 JSP 中，可以使用多个 include 指令来包含多个 JSP 或者 HTML 文件。

由于被引入的文件的内容将在 include 指令的位置被引入，并与引用的文件拼合成为一个文件后再进行编译，所以被引入的文件中不要出现＜html＞＜/html＞＜body＞＜/body＞等 html 标签，以免破坏整个网页的 html 标签结构。

在 Web 应用中，无论页面输出什么内容，页头、页脚和导航栏都是一样的。页头一般是网站的 Logo 图片，而页脚是一些版权的声明等。如果只使用静态网页技术，就需要在这些页面中写入相应的页头、页脚和导航栏的 HTML 代码，这种方式会带来很大的维护问题：一旦这些内容需要进行修改，就需要修改所有的包含了这些内容的页面。通过 JSP 的 include 指令就可以避免这一问题。

课堂练习 4-6

要求用 include 指令在课堂练习 4-1 和 4-2 实现的登录页和注册页中添加导航栏。

要求：登录页和注册页的顶部都显示同样的导航栏，使用导航栏可在这两个页面间自由切换。

提示：将主菜单制作成一个网页，在网站的每个网页中使用 include 指令引入。

首先编写导航栏文件 menu.jsp，如示例 4-10 所示。

示例 4-10 导航栏文件 menu.jsp

```
<%@page pageEncoding="gb18030"%>
<center>
<a href="login.jsp">登录</a>
<a href="register.jsp">注册</a>
```

在 login.jsp 和 register.jsp 中添加如下语句：

示例 4-11　引入导航栏的 login.jsp

```
<%@page  contentType="text/html;charset=GB18030" language="java"%>
<HTML>
  <HEAD><TITLE>登录页</TITLE></HEAD>
  <BODY align="center">
  <%@ include file="menu.jsp"%>
  <table width="316" border="1" align="center" bordercolor="#666666" bgcolor=
  "#CCCCCC">
    <tr>
      <td width="306" bgcolor="#CCCC99">
        <table width="280" align="center" cellpadding="10"><tr align=
        "center"><td>
       <FORM name="form1" method="post" action="loginCheck.jsp"></P>
       <P>用户名<INPUT type="text" name="username">
       <P>密 码 <INPUT type="password" name="password"></P>
       <P><INPUT type="submit" name="submit" value="提交" />
      </P>
  </FORM>
</th></tr></table>
</td></tr></table>
</BODY>
<HTML>
```

示例 4-11 的显示效果如图 4-13 所示。



图 4-13　引入了导航栏的登录页

include 指令的优缺点如下：include 指令可以在转换阶段就将对应的文件包含进来，所以，可以在 JSP 中定义在主页面中使用到的变量等。这是 include 指令的优点。但是，它有一个很大的缺点，当被包含中的文件内容发生改变的时候，服务器可能不能监测到，因此，此时服务器不会对它进行重新转换和编译，这可能会带来很大的维护问题，因为这个时候需要改变所有用到改动的被包含文件的 JSP 文件的修改日期，这样才能够让对应的 JSP 文件重新转换和编译。

4.4　JSP 动作标签

所谓 JSP 动作标签,是在 JSP 中利用 XML 语法格式的标记来控制 Servlet 容器的行为。

动作标签的语法格式为:

```
<jsp:directive {attr=value}>
```

JSP 动作包括如下内容。

- jsp:include　在页面被请求的时候引入一个文件。
- jsp:useBean　获得一个 JavaBean 实例。
- jsp:setProperty　设置 JavaBean 的属性。
- jsp:getProperty　获得某个 JavaBean 的属性。
- jsp:forward　把请求转到一个新的页面。
- jsp:param　为其他标签提供附加信息。
- jsp:plugin　根据浏览器类型为 Java 插件生成 OBJECT 或 EMBED 标记。

本节主要讲解 include 和 forward 这两个动作,而 useBean、setProperty、getProperty 等动作将在第 7 章讲解。plugin 动作在开发中用得很少,不再讲解,请读者根据其他的动作以及相关文档学习。

4.4.1　include 动作

include 动作标签也能起到引入一个文件的作用。与 include 指令标签不同的是:include 动作标签"告诉"JSP 页面动态地包含一个文件,即当 JSP 引擎把 JSP 页面转译成 Java 文件时,不把 JSP 页面中动作指令 include 所包含的文件与原 JSP 页面合并成一个新的 JSP 页面,而是"告诉"Java 解释器,这个文件在 JSP 运行时才包含进来,这样就克服了 include 指令的缺点。

include 指令的语法如下:

```
<jsp:include page="includeFile"/>
```

这是一个 XML 语法结构,所以这个标记要有结束符。它的另外一种语法方式如下:

```
<jsp:include page="includeFile">
</jsp:include>
```

其中,page 属性用于指定需要包含进来的资源,它可以是 HTML 文件、JSP 输出、Servlet 输出等。它只会将被包含文件中的输出包含进来,如果被包含文件中不会产生输出,那么使用这个 include 动作就没有什么意义。

在 include 动作中,除了 page 这个必需的属性以外,还有一个可选的 flush 属性,它可

以取值 true 或者 false。这个属性用于控制是否在页面包含进来之前清空主页面的输出流(默认为 false,表示不清空)。

4.4.2　forward 动作

在 JSP 中,可以使用 forward 动作来实现程序的转向,例如,从 loginCheck.jsp 转向到 welcome.jsp 可以在 loginCheck.jsp 中使用 forward 动作。

```
<jsp:forward page="welcome.jsp"/>
```

这样,当在 loginCheck.jsp 中执行 forward 动作的时候,它将会跳转到 welcome.jsp。

4.4.3　plugin 动作

在页面中使用普通的 HTML 标记<applet>...</applet>,可以让客户端下载并运行一个 Java applet 小应用程序,但并不是所有的客户的浏览器都支持 Java applet 小程序。而使用 plugin 动作标签可以保证客户能执行小应用程序。

<jsp:plugin>为 Web 开发人员提供了一种在 JSP 文件中嵌入客户端运行的 Java 程序的方法。一般来说,<jsp:plugin>元素会指定对象是 Applet 还是 Bean,同样也会指定 class 的名字及位置,另外还会指定将从哪里下载这个 Java 插件。

<jsp:plugin>常用属性及说明示例如下:

```
<jsp:plugin type="applet|bean" code="类文件名" codebase="类文件目录"
            align="bottom|top|middle|left|right" height="插件显示区域的高度"
            width="插件显示区域的高度">
```

4.4.4　param 动作

param 标签以“名字—值”的形式为其他标签提供附加信息。这个标签与 include、forward 和 plugin 动作标签一起使用。

例如下面的用法,可以向跳转到的页面传送信息。

```
<jsp:forward page="desFile">
        <jsp:param name="paramName" value="paramValue"/>
</jsp:forward>
```

4.4.5　JavaBean 相关动作标签

<jsp:useBean>:用来装载一个在 JSP 页面中使用的 JavaBeans。

<jsp:getProperty>:用来获取 Bean 的属性值并将其转化为一个字符串,然后将其插入到输出页面中。

<jsp:setProperty>：用来设置、修改 Bean 中的属性值。

这些标签的使用方法将在第 5 章中结合 JavaBean 的学习再详细讲解。

4.5 内置对象

JSP 的内置对象在 JSP 中非常重要，这些内置对象是由 Web 容器创建出来的，所以用户不用自己创建。JSP 内置对象是 Web 容器加载的一组类，它不像一般的 Java 对象那样用 new 去获取实例，而是可以直接在 JSP 页面中使用的对象。

JSP 一共有 9 个内置对象，如表 4-3 所示。

表 4-3 JSP 内置对象描述

内置对象名	实例化的类	描述
request	javax.servlet.http.HttpServletRequest	该对象封装了一次请求，客户端的请求参数都被封装在该对象里
response	javax.servlet.http.HttpServletResponse	代表服务器对客户端的响应
session	javax.servlet.http.HttpSession	该对象代表一次会话。从客户端浏览器与站点建立连接起即开始会话，直到关闭浏览器时结束会话
application	javax.servlet.servletContext	该实例代表 JSP 所属的 Web 应用本身，可用于 JSP 页面或者在 Servlet 之间交换信息
out	javax.servlet.jsp.jspWriter	该实例代表 JSP 页面的输出流，用于输出内容，从而形成 HTML 页面
pageContext	javax.servlet.jsp.pageContext	该对象代表该 JSP 页面的上下文，使用该对象可以访问页面中的共享数据
page	java.lang.Object	代表该页面本身
config	javax.servlet.servletConfig	该实例代表该 JSP 的配置信息
exception	java.lang.Throwable	该实例代表其他页面中的异常和错误。只有当页面是错误处理页面，即编译指令 page 的 isErrorPage 属性为 true 时，该对象才可以使用

4.5.1 request 对象

客户端的请求信息被封装在 request 对象中，通过它可以了解到客户的需求，然后做出响应。它是 HttpServletRequest 的实例。

常用方法说明如下。

(1) object getAttribute(String name)：返回指定属性的属性值。

(2) Enumeration getAttributeNames()：返回所有可用属性名的枚举。

(3) String getCharacterEncoding()：返回字符编码方式。

(4) int getContentLength()：返回请求体的长度(以字节数)。

(5) String getContentType():得到请求体的 MIME 类型。

(6) ServletInputStream getInputStream():得到请求体中一行的二进制流。

(7) String getParameter(String name):返回 name 指定参数的参数值。

(8) Enumeration getParameterNames():返回可用参数名的枚举。

(9) String[] getParameterValues(String name):返回包含参数 name 的所有值的数组。

(10) String getProtocol():返回请求用的协议类型及版本号。

(11) String getScheme():返回请求用的计划名,如 http、https 及 ftp 等。

(12) String getServerName():返回接受请求的服务器主机名。

(13) int getServerPort():返回服务器接受此请求所用的端口号。

(14) BufferedReader getReader():返回解码过了的请求体。

(15) String getRemoteAddr():返回发送此请求的客户端 IP 地址。

(16) String getRemoteHost():返回发送此请求的客户端主机名。

(17) void setAttribute(String key,Object obj):设置属性的属性值。

(18) String getRealPath(String path):返回一虚拟路径的真实路径。

课堂练习 4-7

在课堂练习 4-6 的基础上,进一步模拟用户登录的功能。

要求:当用户输入指定的用户名和密码时,跳转到欢迎信息页 welcome. jsp;当用户名和密码错误时,跳转到错误提示页 error. jsp。

提示:判断字符串 s1 与 s2 是否相等用 s1. equals(s2),返回值为布尔型 true/false。

示例 4-12　登录处理页 loginCheck. jsp

```
<%
    String uname=request.getParameter("username");
    String pass=request.getParameter("password");

    if(uname!=null&&pass!=null&&uname.equals("liming")&&pass.equals("aaa"))
    {
%>
        <jsp:forward page="welcome.jsp">
            <jsp:param name="fromPage" value="logincheck.jsp"/>
        </jsp:forward>
<%
    }
    else{
%>
    <jsp:forward page="error.jsp"/>
<%
     }
%>
```

使用request内置对象的getParameter方法接收login.jsp页表单元素提交的参数

用户名为liming密码为aaa时跳转到welcome.jsp页

否则跳转到error.jsp页

课堂练习 4-8

在课堂练习 4-6 的基础上，进一步模拟用户注册的功能。

要求：当用户单击注册页上的“注册”按钮时，使用户在注册页上输入的信息能提交。

提示：正确选择 request 内置对象的方法来完成。

示例 4-13　改写后的注册页 register.jsp

```
<%@page  contentType="text/html;charset=GB18030" language="java" import=
"java.util.*"%>
<HTML>
  <HEAD><TITLE>注册页</TITLE></HEAD>
  <BODY>
<%@include file="menu.jsp"%>
<FORM name="form1" method="post" action="">
  <table width="316" border="1" align="center" bordercolor="#666666" bgcolor=
  "#CCCCCC">
    <tr>
      <td width="306" bgcolor="#CCCC99">
    <table width="330"  align="center"  cellpadding="10">
        <tr><td width="73" height="30">用户名</td>
           <td width="185"><input type="text" name="username"></td>
       </tr>
    <tr>
      <td height="30">密   码</td>
      <td><input type="password" name="password"></td>
    </tr>
    <tr>
      <td>性   别</td>
      <td><input type="radio" name="gender" value="男" />男
          <input type="radio" name="gender" value="女" />女 </td>
    </tr>
    <tr>
      <td height="30">自我介绍</td>
      <td><textarea name="introduction" rows="10" clos="36"></textarea></td>
    </tr>
    <tr>
      <td height="30">爱   好</td>
      <td>音乐 <input type="checkbox" name="hobbycheckbox" value="音乐">
          绘画<input type="checkbox" name="hobbycheckbox" value="绘画"></td>
    </tr>
    <tr>
      <td height="30">来   自</td>
      <td><select name="comefrom">
        <option>罗湖区</option>
        <option>福田区</option>
        <option>南山区</option>
        <option>盐田区</option>
        <option>龙岗区</option>
```

```
        <option>宝安区</option>
      </select></td>
    </tr>
    <tr>
    <td colspan="2" align="center"><input type="submit" name="submit" value=
    "提交"></td>
    </tr>
  </table>
  </td>
  </tr>
  </table>

    </FORM>
<%
   String username=request.getParameter("username");
   String password=request.getParameter("password");
   String gender=request.getParameter("gender");
   String introduction=request.getParameter("introduction");
   String comefrom=request.getParameter("comefrom");
   if(request.getParameterValues("hobbycheckbox") !=null)
   {
         String[] ch1=request.getParameterValues("hobbycheckbox");
         for(int i=0;i<ch1.length;i++){
             out.println(ch1[i]+"<br>");
         }
     }
%>
  </BODY>
<HTML>
```

使用request内置对象的getParameterValues方法接收register.jsp页复选框的参数

课堂练习 4-9　获取客户端的 IP 地址。

要求：用户在访问网站时，在导航栏中显示的欢迎信息中含有用户的 IP 地址。

示例 4-14　改写后的导航栏页 menu.jsp

```
<%@page pageEncoding="gb18030"%>
<center>你好来自<%=request.getRemoteAddr()%>的朋友</center><br>


<table width="850" align="center" bgcolor==#ff6633>
  <th align="center">
    <td align="left">
<a href="/4.2/login.jsp">登录</a>
<a href="/4.2/register.jsp">注册</a>
  </th>
</table>
```

使用request内置对象的getRemoteAddr方法获取客户端的IP地址

4.5.2 response 对象

response 对象包含了响应客户请求的有关信息，它将浏览器参考信息，如回应的 heade、网页采用的编码集等提供给客户端，但在 JSP 中很少直接用到它。它是 HttpServletResponse 方法的实例。

常用方法说明如下。

(1) String getCharacterEncoding()：返回应用的字符编码。

(2) ServletOutputStream getOutputStream()：返回响应的一个二进制输出流。

(3) PrintWriter getWriter()：返回可以向客户端输出字符的一个对象。

(4) void setContentLength(int len)：设置响应头长度。

(5) void setContentType(String type)：设置响应的 MIME 类型。

(6) void sendRedirect(java.lang.String location)：重新定向客户端的请求。

response.sendRedirect()与<jsp:forward>的区别如下。

- 调用 sendRedirect()方法的重定向访问过程结束后，浏览器地址栏中显示的 URL 会改变；而使用 forward 动作不会改变。
- sendRedirect 方法响应的结果就是告诉浏览器去重新发出对另外一个 URL 的访问请求(request)。使用 forward 动作标签跳转是在服务器端内部将请求转发给另外一个资源，所以跳转前后属同一个 request 作用域。

4.5.3 session 对象

从一个客户打开浏览器并连接到服务器开始，到客户关闭浏览器离开这个服务器结束，被称为一个会话(session)。当一个客户访问一个服务器时，可能会在这个服务器的几个页面之间反复连接、反复刷新一个页面或不断地向一个页面提交信息等，服务器应当通过某种办法知道这是一个客户，这就需要 session 对象。

session 对象用来保存某个特定客户端一次访问的一些特定信息。Session 对象在第一个 JSP 页面被装载时自动创建，完成会话期管理。当一个客户端向服务器发送第一个请求时，服务器为其建立一个 session，并为此 session 创建一个标识号。这个用户随后的所有请求都应包括这个标识号。服务器会校对这个标识号以判断请求属于哪个 session。这种机制不使用 IP 作为标识，是因为很多机器是通过代理服务器方式上网，没法区分每一台机器。当用户关闭浏览器并释放 session，或者服务器检测到 session 非活跃状态已经超时，对应该客户端的 session 对象将失效，客户端与服务器的会话关系消失。

session 是 javax.Servlet.HttpSession 的实例。

session 常用方法如下。

(1) long getCreationTime()：返回 session 创建的时间。

(2) public String getId()：返回 session 创建时 JSP 引擎为它设置的唯一 ID 号。

(3) long getLastAccessedTime()：返回此 session 里客户端最近一次请求的时间。

(4) int getMaxInactiveInterval()：返回两次请求间隔多长时间此 session 被取消。

(5) String[] getValueNames()：返回一个包含此 session 中所有可用属性的数组。

(6) void invalidate()：取消 session，使 session 不可用。

(7) boolean isNew()：返回服务器创建的一个 session，确认客户端是否已经加入。

(8) void removeValue(String name)：删除 session 中指定的属性。

(9) void setMaxInactiveInterval(int seconds)：设置两次请求间隔多长时间此 session 会被取消登录。

课堂练习 4-10　用户会话追踪

要求：在课堂练习 4-8 的基础上，当用户正确登录后，在导航栏上显示欢迎×××的信息；并出现"退出"功能超链接，当单击"退出"时退出追踪并显示已退出的提示。

示例 4-15　添加了记录用户登录状态后的 loginCheck.jsp

```
<%
    String uname=request.getParameter("username");
    String pass=request.getParameter("password");
    if(uname!=null&&pass!=null&&uname.equals("liming")&&pass.equals("aaa"))
    {
        session.setAttribute("user",uname);
%>
        <jsp:forward page="welcome.jsp">
          <jsp:param name="fromPage" value="logincheck.jsp"/>
        </jsp:forward>
<%
   }
   else{
%>
         <jsp:forward page="error.jsp"/>
<%
   }
%>
```

使用session的setAttribute方法在session中记录用户的登录状态。在session存储一个名字为user的用户的一个属性

示例 4-16　判断用户是否登录 menu.jsp

```
<%@page pageEncoding="gb2312"%>
<center>你好来自<%=request.getRemoteAddr()%>的朋友</center><br>
<table width="850" align="center" bgcolor==#ff6633>
  <th align="center">
    <td align="left">

<a href="/login.jsp">登录</a>
<a href="/register.jsp">注册</a>
  <%
   if(session.getAttribute("acc")!=null)
   {
   %>
```

使用session的getAttribute方法判断用户是否已登录，并获取已登录用户的用户名

```
      <td align=right width=200>用户<%=session.getAttribute("acc")%>已登录</td>
      <td align=center width=100><a href='logout.jsp'>退出<td>
    <%}
    %>
    </th>
  </table>
```

已登录用户单击导航栏中的“退出”超链接，将实现退出登录状态的功能。

示例 4-17　退出登录页 logout.jsp

```
<%@ page  contentType="text/html; charset=gb2312" language="java" import=
"java.util.*"%>
<HTML>
  <HEAD><TITLE>登出页</TITLE></HEAD>
  <BODY align="center">
<%
   session.invalidate();
%>
  <table width="360" align="center">
    <tr><td align="center">欢迎下次再来!</td></tr>
    <tr><td align="center"><a href="login.jsp">返回登录页</a></td></tr>
  </table>
  </BODY>
<HTML>
```

使用session的invalidate方法使session失效，从而达到用户退出系统的目的

4.5.4　application 对象

在服务器启动后，会产生一个 application 对象。当一个客户访问服务器上的一个 JSP 页面时，JSP 引擎为客户分配这个 application 对象。当客户在所访问的网站的各个页面之间浏览时，使用的 application 对象都是同一个，直到服务器关闭，这个 application 对象才被取消。

在使用 session 时，各个客户独享各自的 session 对象，而使用 application 时，客户在同一个服务器中共享一个 application 对象，因此，application 对象可以用来保存服务器中一些公共的数据。

application 是 javax.servlet.ServletContext 的实例。

application 常用方法如下。

(1) Object getAttribute(String name)：返回给定对象的属性值。

(2) Enumeration getAttributeNames()：返回所有可用属性名的枚举值。

(3) void setAttribute(String name,Object obj)：设定对象的属性值。

(4) void removeAttribute(String name)：删除属性及属性值。

(5) String getServerInfo()：返回 JSP(SERVLET)引擎名及版本号。

(6) String getRealPath(String path)：返回一虚拟路径对应的真实路径。

(7) ServletContext getContext(String uripath)：返回指定 WebApplication 的

application 对象。

(8) int getMajorVersion()：返回服务器支持的 Servlet API 的最大版本号。

(9) int getMinorVersion()：返回服务器支持的 Servlet API 的最小版本号。

(10) String getMimeType(String file)：返回指定文件的 MIME 类型。

(11) URL getResource(String path)：返回指定资源(文件及目录)的 URL 路径。

(12) InputStream getResourceAsStream(String path)：返回指定资源的输入流。

(13) RequestDispatcher getRequestDispatcher(String uripath)：返回指定资源的 RequestDispatcher 对象。

(14) Servlet getServlet(String name)：返回指定名的 Servlet。

(15) Enumeration getServlets()：返回所有 Servlet 的枚举值。

(16) Enumeration getServletNames()：返回所有 Servlet 名的枚举值。

(17) void log(String msg)：把指定消息写入 Servlet 的日志文件。

(18) void log(Exception exception,String msg)：把指定异常的栈轨迹及错误消息写入 Servlet 的日志文件。

(19) void log(String msg,Throwable throwable)：把栈轨迹及给出的 Throwable 异常的说明信息写入 Servlet 的日志文件。

课堂练习 4-11　聊天室

要求：在课堂练习 4-8 的基础上，当用户正确登录后，在导航栏上显示欢迎×××的信息，并出现“退出”功能超链接，当单击该超链接时退出追踪并显示已退出的提示。

示例 4-18　聊天室 chat.jsp

```
<%@page  contentType="text/html;charset=GB18030" language="java" import=
"java.util.*"%>

<HTML>
  <HEAD><TITLE>聊天室</TITLE></HEAD>
  <BODY>
  <jsp:include page="menu.jsp"/>
<%
if(session.getAttribute("acc")!=null)
{
  response.setHeader("Refresh","15");
  String userName=(String)session.getAttribute("acc");
  request.setCharacterEncoding("GB2312");
  String chat=request.getParameter("mychat");

  String chats=(String)application.getAttribute("chat");

  if(chat!=null)
  {
    Date date=new Date();
```

```
        chat=userName+"("+date.toString()+")"+chat;
        if(chats==null)
          chats=chat;
        else
          chats=chats+"\r\n"+chat;
        chat=null;
      }
      if(chats!=null)
      {
       application.setAttribute("chat",chats);
       out.print("<table border=2 width=400 bordercolorlight=#FFFFFF
       bordercolordark=#000000 cellspacing=0 cellpadding=0 align=center>");
       out.print("<tr>");
       out.print("<th>聊天室</th>");
       out.print("</tr>");
       out.print("<tr>");
       out.print("<td>");
       out.print("<textarea name=introduction rows=10 clos=80 wrap=physical style
       =width:400>"+application.getAttribute("chat"));
       out.print("</textarea>");
       out.print("</td>");
       out.print("</tr>");
       out.print("</table>");
      }
      out.print("<table width=400 cellspacing=0 cellpadding=0 align=center>");
      out.print("<tr height=30>");
      out.print("<th>");
      out.print("<Form action=chat.jsp method=post>");
      out.print("<input type=text size=30 name=mychat>");
      out.print("<input type=submit name=submit value=发言>");
      out.print("</Form>");
      out.print("</th>");
      out.print("</tr>");
      out.print("</table>");
    }
    else
    {
      out.print("<table width=400 align=center>");
      out.print("<tr>");
      out.print("<th><a href='login.jsp'>请登录</a></th>");
      out.print("</tr>");
      out.print("<tr>");
      out.print("</table>");
    }
    %>
      </BODY>
    <HTML>
```

4.5.5　out 对象

out 对象主要用来向客户端输出各种格式的数据，并且管理应用服务器上的输出缓冲区。out 是 javax.servlet.JspWriter 的实例。

out 对象常用方法如下。

(1) newLine()：用于输出一个换行符。

(2) void flush()：强制输出服务器中的数据。如果预编译指令中 page 的 autoFlush 属性值设置为 true，那么 JSP 程序会把输出数据缓存在服务器的缓冲区里，直到程序结束或者缓冲区充满了数据，服务器才会自动把缓冲区中的数据输出到客户端。如果在 JSP 程序里使用了 flush()方法，那么服务器不管缓冲区是否已经充满值，都将数据输出到客户端。如果预编译指令中 page 的 autoFlush 属性值设置为 false，那么需要显式调用 flush 将数据输出到客户端。

(3) void close()：该方法首先将缓冲区里的数据输出到客户端，然后关闭对客户端的输出流。

(4) void clearBuffer()：该方法用于清除缓冲区里的数据，并且把数据写到客户端。在缓冲区的数据为空的时候，这个方法将会产生 IOException 错误。

(5) void clear()：该方法用于清除缓冲区里的数据，但不把数据写到客户端。在缓冲区的数据为空的时候，这个方法将会产生 IOException 错误，所以一般要使用 try...catch...块包住。

(6) int getBufferSize()：该方法可以获取缓冲区的大小。缓冲区的大小是通过预编译指令 page 和 buffer 属性来确定的。

(7) int getRemaining()：该方法可以获得缓冲区没有使用的字节数目。

(8) Boolean isAutoFlush()：该方法返回布尔值，返回值由 page 指令的 autoFlush 属性值决定。

4.5.6　page 对象

page 对象就是指向当前 JSP 页面本身，它是 java.lang.Object 的实例。

pape 对象常用方法如下。

(1) class getClass：返回一个对象的类。

(2) int hashCode()：返回一个对象的 hash 码。

(3) boolean equals(Object obj)：判断一个对象是否与指定的其他对象相等。

(4) void copy(Object obj)：把一个对象复制到指定的对象中。

(5) Object clone()：克隆一个对象。

(6) String toString()：把一个对象转换成 String 类的对象。

(7) void notify()：唤醒一个等待的线程。

(8) void notifyAll()：唤醒所有等待的线程。

(9) void wait(int timeout):使一个线程处于等待直到 timeout 结束或被唤醒。

(10) void wait():使一个线程处于等待,直到被唤醒。

(11) void enterMonitor():对一个对象加锁。

(12) void exitMonitor():对一个对象开锁。

4.5.7 exception 对象

exception 对象是一个例外对象,当一个页面在运行过程中发生了例外,就产生了这个对象。如果一个 JSP 页面要应用此对象,就必须把 isErrorPage 设为 true,否则无法编译程序。exception 对象是 java.lang.Throwable 的实例。

exception 对象常用方法如下。

(1) String getMessage():返回描述异常的消息。

(2) String toString():返回关于异常的简短描述消息。

(3) void printStackTrace():显示异常及其栈轨迹。

(4) Throwable FillInStackTrace():重写异常的执行栈轨迹。

4.5.8 pageContext 对象

pageContext 对象提供了对 JSP 页面内所有对象及名字空间的访问,可以访问本页所在的 session,也可以读取本页面所在的 application 对象的某一属性值,相当于页面中所有功能的集大成者,它是 javax.servlet.jsp.PageContext 的实例。

pageContext 对象常用方法如下。

(1) JspWriter getOut():返回当前客户端响应被使用的 JspWriter 流(out)。

(2) HttpSession getSession():返回当前页中的 HttpSession 对象(session)。

(3) Object getPage():返回当前页的 Object 对象(page)。

(4) ServletRequest getRequest():返回当前页的 ServletRequest 对象(request)。

(5) ServletResponse getResponse():返回当前页的 ServletResponse 对象(response)。

(6) Exception getException():返回当前页的 Exception 对象(exception)。

(7) ServletConfig getServletConfig():返回当前页的 ServletConfig 对象(config)。

(8) ServletContext getServletContext():返回当前页的 ServletContext 对象(application)。

(9) void setAttribute(String name,Object attribute):设置属性及属性值。

(10) void setAttribute(String name,Object obj,int scope):在指定范围内设置属性及属性值。

(11) public Object getAttribute(String name):取属性的值。

(12) Object getAttribute(String name,int scope):在指定范围内取属性的值。

(13) public Object findAttribute(String name):寻找一属性,返回其属性值或

NULL。

(14) void removeAttribute(String name)：删除某属性。

(15) void removeAttribute(String name,int scope)：在指定范围内删除某属性。

(16) int getAttributeScope(String name)：返回某属性的作用范围。

(17) Enumeration getAttributeNamesInScope(int scope)：返回指定范围内可用的属性名枚举。

(18) void release()：释放 pageContext 所占用的资源。

(19) void forward(String relativeUrlPath)：使当前页面重导到另一页面。

(20) void include(String relativeUrlPath)：在当前位置包含另一文件。

4.5.9　config 对象

config 对象是 javax.servlet.ServletConfig 的实例，是在一个 Servlet 初始化时，JSP 引擎向它传递信息用的，此信息包括 Servlet 初始化时所要用到的参数（通过属性名和属性值构成）以及服务器的有关信息（通过传递一个 ServletContext 对象）。

config 对象常用方法如下。

(1) ServletContext getServletContext()：返回含有服务器相关信息的 ServletContext 对象。

(2) String getInitParameter(String name)：返回初始化参数的值。

(3) Enumeration getInitParameterNames()：返回 Servlet 初始化所需所有参数的枚举。

4.6　JDBC 简 介

4.6.1　JDBC 的概念及特点

JDBC(Java Database Connectivity，Java 数据库连接）是一种用于执行 SQL 语句的 Java API，可以为多种关系数据库提供统一的访问接口。JDBC 由一组用 Java 语言编写的类与接口组成，通过调用这些类和接口所提供的方法，用户能够以一致的方式连接多种不同的数据库系统（如 Access、Server 2000、Oracle、Sybase 等），进而使用标准的 SQL 语言来存取数据库中的数据，而不必再为每一种数据库系统编写不同的 Java 程序代码。

简单地说，JDBC 能完成下列三个功能：

- 与一个数据库建立连接；
- 向数据库发送 SQL 指令；
- 处理数据库返回的结果。

4.6.2 Web 访问数据库的原理

Web 访问数据库的一般流程如图 4-14 所示，客户端的请求以 JSP 或者 HTML 的形式发送到服务器，经 Web 服务器解析后，对于需要进行数据库访问的业务逻辑，Web 服务器将通过 JDBD 的形式连接数据库服务器（如 Oracle、SQL Server、MySQL 等），数据库服务器执行数据表中记录的增、删、改、查等操作，并将结果返回给 Web 服务器，Web 服务器再将操作数据库的结果以 HTML 的形式通过 HTTP 协议回传给前端的浏览器。

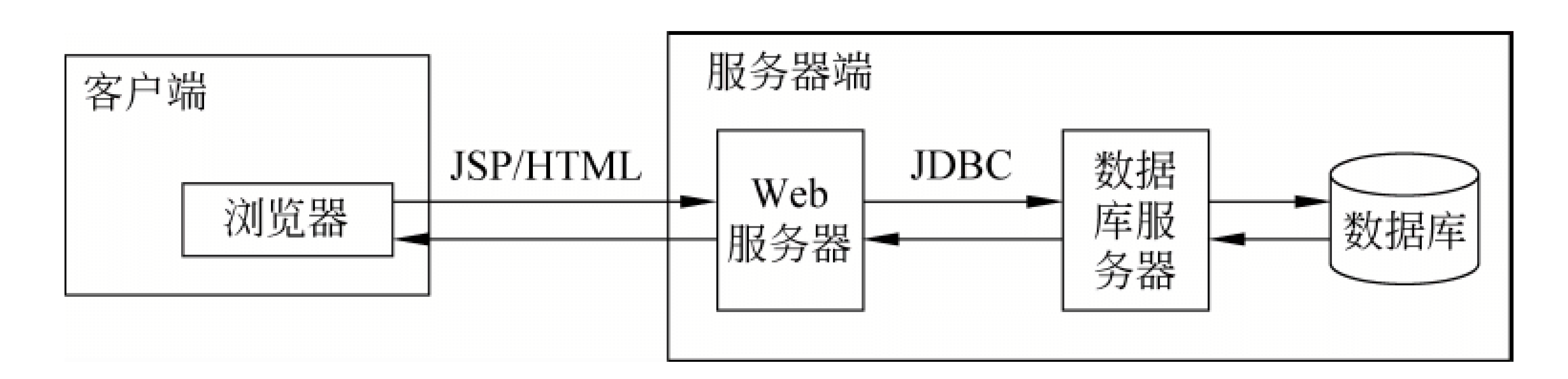


图 4-14　Web 操作数据库流程图

4.6.3 JDBC 的结构

JDBC API 通过一个数据库管理器（Database Manager）和为各种数据库定制的驱动程序提供与不同数据库的透明连接。JDBC 数据库管理器将确保正确的驱动程序被用于连接数据源。它可以同时支持与不同数据库的连接。

JDBC 数据库管理器将标准的 JDBC 指令转换成适用于不同数据库通信的网络协议指令或其他 API 指令。这种指令的转换机制，使基于 JDBC 接口开发的程序可以独立于数据库的种类。如果底层的数据库被更换了，用户只需相应的替换程序中所引用的 JDBC 驱动程序即可。

4.6.4 JDBC 的种类

数据库连接对动态网站来说是最重要的部分。很多数据库系统带有 JDBC 驱动程序，通过 JDBC 驱动程序与数据库相连，可以执行查询和提取数据等操作。目前，JDBC 的驱动程序主要有以下四种。

1. JDBC-ODBC 桥

驱动程序用于连接 JDBC 和另一种数据库连接机制 ODBC，JDBC-ODBC 桥使基于 JDBC 的程序能通过传统的 ODBC 驱动程序访问数据库。由于大多数数据库系统都带有 ODBC 驱动程序，所以 Java 程序能访问诸如 Oracle、Sybase、MS SQL Server 和 MS Access 等数据库。这一驱动程序已被包括在 Java 2 SDK 的 sun.jdbc.odbc 包中。这种方法要求客户端必须安装 ODBC 驱动，所以不适合 Web 应用。

2. 本地 API 驱动

本地 API 驱动直接把 JDBC 调用转变为数据库的标准调用再去访问数据库。这种驱动比 JDBC-ODBC 桥执行效率高，但是，仍然需要在客户端加载数据库厂商提供的类库，因此，同样不适合 Web 应用。

3. Java 到网络协议

这类驱动程序通过一定的网络协议与数据库服务器上的 JDBC 中间件(Middleware)通信。这一中间件程序将网络协议指令转换成数据库指令。此类驱动程序不需要在客户端安装目标数据库的类库。

4. Java 到数据库协议

此类驱动通过实现一定的数据库协议直接与数据库通信。与 Java 到网络协议的驱动程序一样，它直接与数据库通信，所以其效率是 4 种驱动程序中最高的。其缺点是：当目标数据库的类型更换时，必须重新购买 JDBC 驱动程序并在本地安装。

究竟选择哪一类型的驱动程序，完全取决于所构建 Web 应用的需求。如果不希望在程序客户端安装大量目标数据库的类库，则应该考虑后两种驱动程序；如果后两种驱动程序的价格特别高，那么可以使用前两种驱动程序。

4.6.5　手动建立 ODBC 数据源

下面以一个实例来说明如何使用 JDBC 连接数据库，实例中为便于操作，采用了 JDBC-ODBC 桥的方式连接 Access 数据库。

课堂练习 4-12　JSP 读取数据库的信息

要求：创建一个 Access 数据库 db.mdb，在数据库中创建一个数据表 student，包含的字段如表 4-4 所示。编写 JSP 文件，查询数据表中所有"软件学院"的学生并显示在浏览器中。

表 4-4　数据表中的字段

字段	类型	说明
id	自动编号	序号，主键
name	文本	姓名
school	文本	所属学院
age	整型	年龄

采用这种方式连接数据库，需要在系统中设置 ODBC 数据源，在 Windows 系统中设置数据源的步骤如下。

(1) 进入 ODBC 数据源环境

打开 Windows 中的"控制面板"，找到"管理工具"中的"数据源(ODBC)"工具图标，

如图 4-15 所示。



图 4-15 设置数据源

双击“数据源(ODBC)”，打开“ODBC 数据源管理器”对话框，选择“系统 DSN”选项卡，单击右侧的“添加”按钮，如图 4-16 所示。

提示：在这里也可以选择“用户 DSN”选项卡。“系统 DSN”和“用户 DSN”在使用上的区别是：“用户 DSN”上设置的 DSN 只能在使用进行设置操作的用户名登录时才可以使用；而“系统 DSN”则针对任何登录的用户都可以使用。

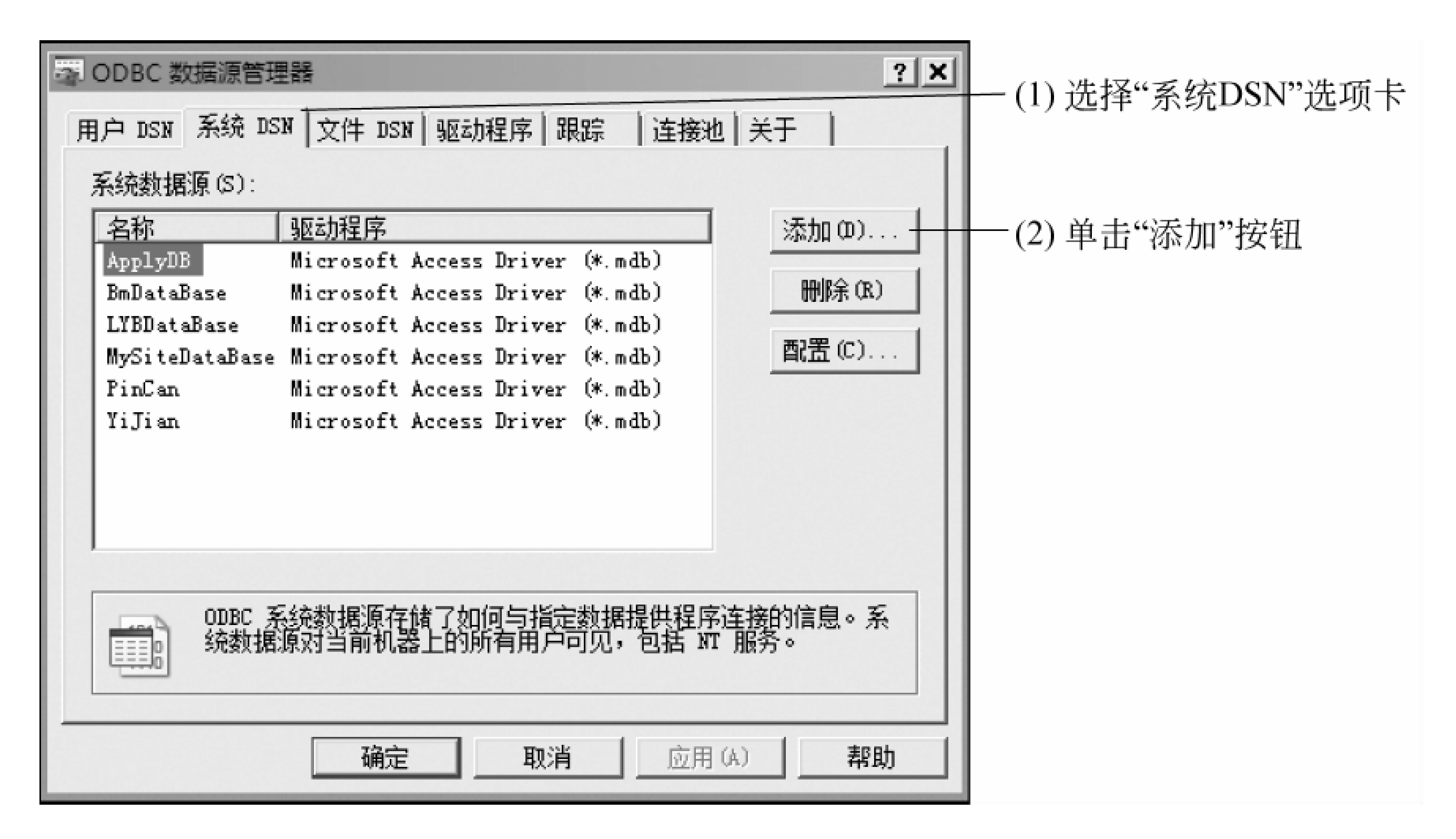


图 4-16 ODBC 数据源管理器

(2) 选择数据源的驱动程序类型

在出现的“创建新数据源”对话框中选择 Access 数据库的驱动程序“Microsoft Access Driver(*.mdb)”，再单击“完成”按钮，如图 4-17 所示。

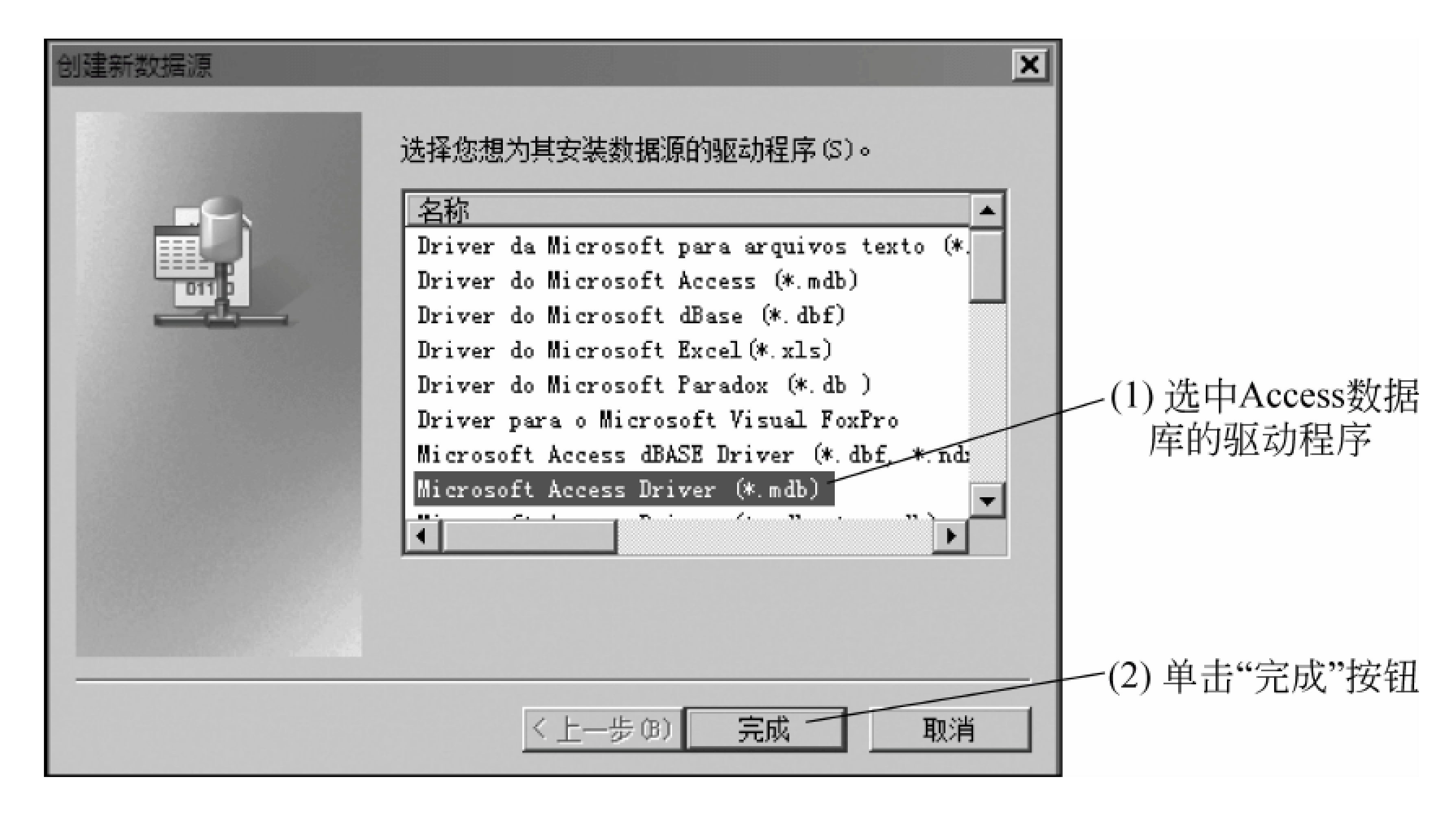


图 4-17 ODBC 数据源管理器

（3）为新数据源命名，并与指定的数据库关联

在出现的“ODBC Microsoft Access 安装”对话框中输入数据源的名称，然后单击“选择”按钮，如图 4-18 所示。



图 4-18　输入数据源名称

在出现的“选择数据库”对话框中，选择数据库所在的文件夹后，数据库会自动出现在左侧列表框中，再单击数据库名称，可以使之出现在上方栏中，然后单击“确定”按钮，如图 4-19 所示。

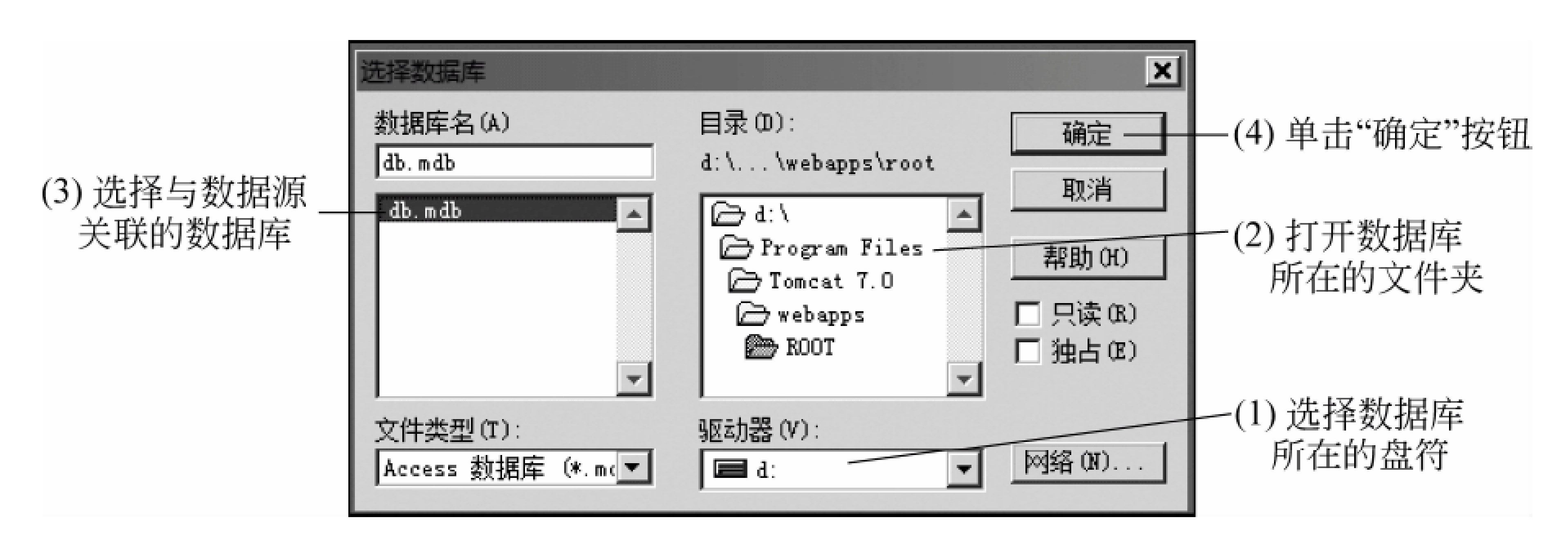


图 4-19　指定数据库

现在返回到“ODBC Microsoft Access 安装”对话框，这时可以看到数据源的名称和与之关联的数据库，单击“确定”按钮，如图 4-20 所示。



图 4-20　完成数据源名称与数据库的关联

接着返回到“ODBC 数据源管理器”对话框，这时可以看到新设置的数据源的名称已经出现在数据源列表中，单击“确定”按钮，完成数据源的设置，如图 4-21 所示。

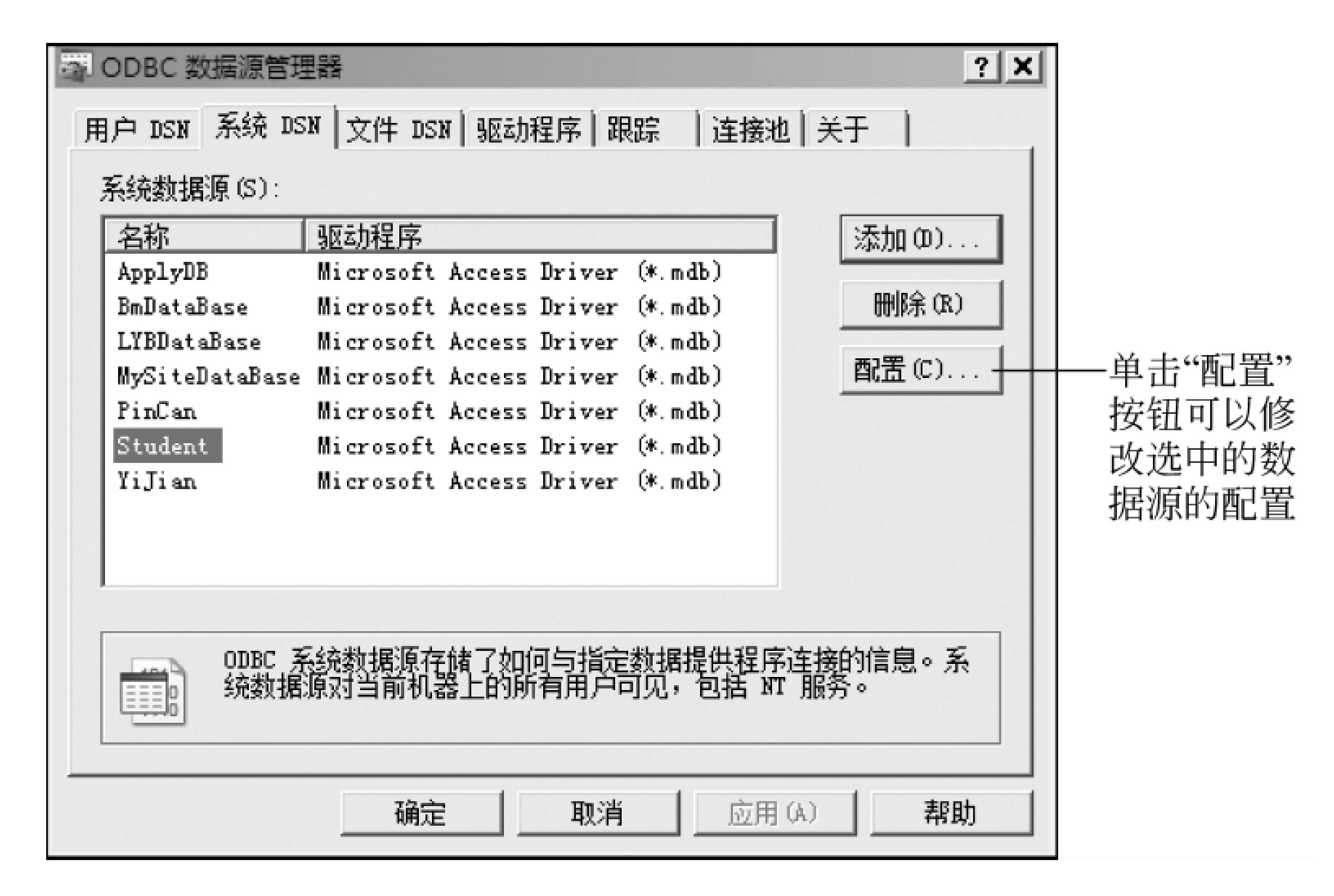


图 4-21　完成数据源的设置

如果是在用户 DSN 选项卡中进行以上设置，还应继续设置用户的登录名称和密码，方法如下。

在建立了数据源名称与数据库的关联后，还要单击“ODBC Microsoft Access 安装”对话框中的“高级”按钮，为数据源设置一个登录名称和密码，如图 4-22 和图 4-23 所示。

图 4-22　设置数据源的高级属性

设置好 Access 数据库的数据源之后，这个数据源 Student 就对应着一个 Access 数据库，即 student.mdb。用户在编写 Java 程序时，就可以将这个数据源作为数据库的连接对象包含在源程序中，以便对存储在数据库中的数据进行存取操作。

说明：对于不同类型的数据库，在建立 ODBC 数据源时，应选择相应的数据库驱动程序。

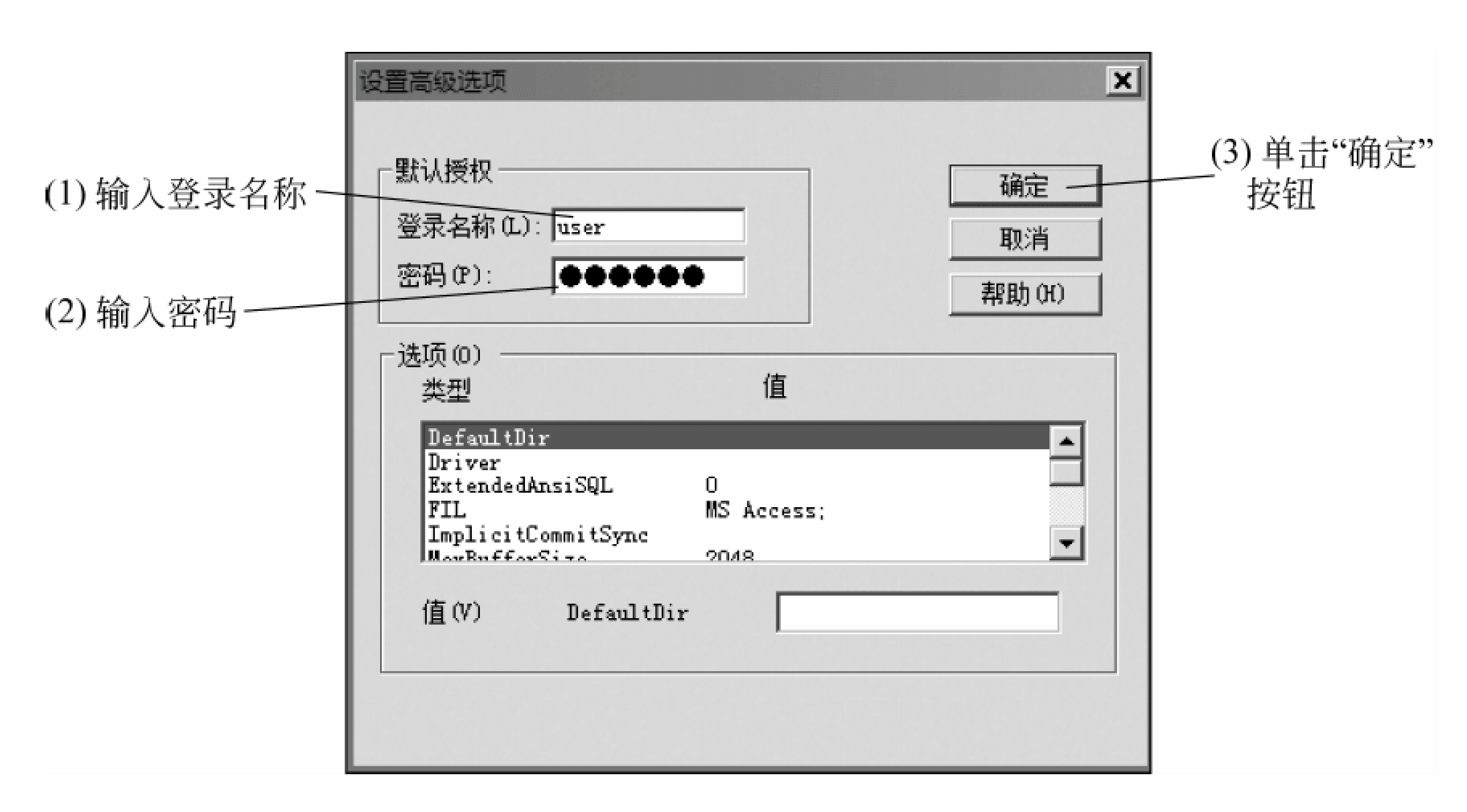


图 4-23　设置登录名称与密码

4.6.6　JDBC 访问数据库的基本步骤

JDBC 提供了 Java 应用程序与各种不同数据库之间进行对话的接口。

使用 JDBC 访问数据库的步骤如下：

第一步　引入 java sql 包

```
<%@page import="java.sql.* " %>
```

第二步　加载 JDBC 驱动程序

在与某一特定数据库建立连接之前，必须首先加载一种可用的 JDBC 驱动程序。这需要使用下列方法来加载 JDBC 驱动程序：

```
Class.forName("DriverName");
```

使用 Class 类的 forName()方法来装载数据库驱动类，并且进行类的初始化等操作。DriverName 是要加载的 JDBC 驱动程序名称。驱动程序名称根据数据库厂商提供的 JDBC 驱动程序的种类来确定。对于 JDBC-ODBC 桥，加载 JDBC-ODBC 数据库驱动程序的方法为：

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

第三步　创建数据库连接

在使用 JDBC 操作数据库之前，必须首先建立一个数据库连接，创建和指定数据库的连接需要使用 DriverManager 类的 getConnection()方法，其一般的使用格式如下：

```
Connection conn = DriverManager. getConnection (String url, String user, String
password);
```

该方法返回的是一个 java. sql. Connection 对象。这里的 URL 是一个字符串，代表了将要连接的数据源，即具体的数据库位置。不同的 JDBC 驱动程序其 URL 格式是不同

的。通过设置数据源的方式连接 4.6.6 小节中 student.mdb 数据库的代码示例如下：

```
Class.forName("sun.jdbc.ordc.JdbcOdbcDriver");
String user="";
String password="";
Connection Conn = DriverManager. getConnection ( " jdbc: odbc: Student ", user,
password);
```

数据源名称

第四步 创建 Statement 或者预编译 Statement 对象

在与某个特定数据库建立连接之后，这个连接会话就可以用于发送 SQL 语句。在发送 SQL 语句之前，必须创建一个 Statement 类的对象，该对象负责将 SQL 语句发送给数据库。如果 SQL 语句运行后产生结果集，Statement 对象会将结果集返回给一个 ResultSet 对象。

（1）创建 Statement 对象的方法

创建 Statement 对象是使用 Connection 接口的 createStatement()方法来实现的：

```
Statement smt=conn.createStatement();
```

还可以使用预编译 statement 的方式来向数据库发送 SQL 语句和接受结果集。

预编译语句 PreparedStatement 是 java.sql 中的一个接口，它是 Statement 的子接口。通过 Statement 对象执行 SQL 语句时，需要将 SQL 语句发送给数据库管理系统（DataBase Management System，DBMS），由 DBMS 进行编译后再执行。预编译语句和 Statement 不同，在创建 PreparedStatement 对象时就指定了 SQL 语句，该语句立即发送给 DBMS 进行编译。当该编译语句被执行时，DBMS 直接运行编译后的 SQL 语句。

（2）预编译 SQL 语句的方法

① 创建 PreparedStatement 对象

```
PreparedStatement psmt= conn.prepareStatement ("SELECT * FROM employee WHERE
name=?AND password=?");
```

创建包含带两个 IN 参数占位符“?”的 SQL 语句的 PreparedStatement 对象。

② 传递 IN 参数

在执行 PreparedStatement 对象之前，必须设置每个“?”参数的值。这可通过调用 setXXX()方法来完成，其中 XXX 是与该参数相应的类型。以下代码将第一个参数设为"liming"，第二个参数设为"123456"：

```
pstmt.setString (1, "liming");
pstmt.setString(2, "123456");
```

一旦设置了给定语句的参数值，其值将一直保留，直到被设置为新值或者调用 clearParameters()方法清除它为止。

（3）使用预编译语句的优点

① 提高效率。当需要对数据库进行数据插入、更新或者删除的时候，程序会发送整个 SQL 语句给数据库处理和执行。数据库处理一个 SQL 语句，需要完成解析 SQL 语句、检查语法和语义以及生成代码；一般说来，处理时间要比执行语句所需要的时间长。

预编译语句在创建的时候已经是将指定的 SQL 语句发送给了 DBMS，完成了解析、检查、编译等工作。因此，当一个 SQL 语句需要执行多次时，使用预编译语句可以减少处理时间，提高执行效率。

② 提高安全性。可以防止恶意的 SQL 语句注入。

```
String sql="SELECT * FROM employee WHERE name=' "+username+" ' AND password=
                    '"+password+" ' ";
```

如果把" ' "作为 username 的参数以及"'OR true"作为 password 传入进来，SELECT 语句就变成：

```
SELECT * FROM employee WHERE name='' AND password='' OR true
```

条件部分的布尔运算结果为 true，所以任意的用户名和密码都可以通过验证。而如果使用预编译语句，传入的任何参数内容不会和原来的语句发生任何匹配的关系，所以使用预编译可以防止恶意的 SQL 注入。

当语句格式固定时，应使用 PreparedStatement；当语句格式无法预见时，才考虑采用 Statement。

第五步　执行 SQL 语句

Statement 对象创建好之后，就可以使用该对象的 executeQuery()方法来执行数据库查询语句。executeQuery()方法返回一个 ResultSet 类的对象，它包含了 SQL 查询语句执行的结果。例如：

```
ResultSet rs=smt.executeQuery("SELECT * from student WHERE school='软件学院'");
```

当使用预编译 PreparedStatement 的方式时，仍然使用该对象的 executeQuery()方法来执行数据库查询语句，但由于 SQL 语句在前面的步骤中已经“准备好了”，所以 PreparedStatement 的 executeQuery()方法不需要 SQL 语句作为参数，执行的结果仍然是一个包含了 SQL 查询语句执行结果 ResultSet 类的对象，示例如下：

```
ResultSet rs=psmt.executeQuery();
```

第六步　接收并处理 SQL 的返回结果

JDBC 接收结果是通过 ResultSet 类的对象来实现的。一个 ResultSet 对象包含了执行某个 SQL 语句后满足条件的所有的行，它还提供了对这些行的访问，用户可以通过一组 getters 方法来访问当前行的不同列。通常结果集的形式是一张带有表头和相应数据的表。

(1) ResultSet 的 getters 方法

① BigDecimal getBigDecimal (int columnIndex)：以具有全部精度的 java.math.BigDecimal 对象形式获取当前行中某个列的值。Palm OS 的 DB2 Everyplace JDBC 驱动程序不支持此方法。

② BigDecimal getBigDecimal(int columnIndex , int scale)：以 Java 编程语言中的 java.math.BigDecimal 对象形式获取此 ResultSet 对象当前行中指定列的值。Palm OS

的 DB2 Everyplace JDBC 驱动程序不支持此方法。

③ BigDecimal getBigDecimal (String columnName)JDBC 2.0：以具有全部精度的 java.math.BigDecimal 对象形式获取当前行中某个列的值。Palm OS 的 DB2 Everyplace JDBC 驱动程序不支持此方法。

④ BigDecimal getBigDecimal(String columnName, int scale)：以 Java 编程语言中的 java.math.BigDecimal 对象形式获取此 ResultSet 对象当前行中指定列的值。Palm OS 的 DB2 Everyplace JDBC 驱动程序不支持此方法。

⑤ Blob getBlob (int columnIndex)：获取此 ResultSet 对象的当前行中的 BLOB 值。

⑥ Blob getBlob(String columnName)：获取此 ResultSet 对象的当前行中的 BLOB 值。

⑦ boolean getBoolean (int columnIndex)：以 Java 布尔值形式获取当前行中指定列的值。

⑧ boolean getBoolean (String columnName)：以 Java 布尔值形式获取当前行中指定名称列的值。

⑨ byte getByte (int columnIndex)：以 Java 编程语言中的字节形式获取此 ResultSet 对象当前行中指定序号列的值。

⑩ byte getByte (String columnName)：以 Java 编程语言中的字节形式获取此 ResultSet 对象当前行中指定名称列的值。

⑪ byte[] getBytes(int columnIndex)：以 Java 编程语言中的字节数组形式获取此 ResultSet 对象当前行中指定序号列的值。

⑫ byte[] getBytes (String columnName)：以 Java 编程语言中的字节数组形式获取此 ResultSet 对象当前行中指定名称列的值。

⑬ int getConcurrency()：JDBC 2.0：返回结果集的并行性方式。

⑭ Date getDate (int columnIndex)：以 Java 编程语言中的 java.sql.Date 对象形式获取此 ResultSet 对象当前行中指定序号列的值。

⑮ Date getDate(int columnIndex, Calendar cal)：以 Java 编程语言中的 java.sql.Date 对象形式返回此 ResultSet 对象的当前行中指定序号列的值。

⑯ Date getDate(String columnName)：以 Java 编程语言中的 java.sql.Date 对象形式获取此 ResultSet 对象的当前行中指定名称列的值。

⑰ double getDouble(int columnIndex)：以 Java 双精度形式获取当前行中指定序号列的值。

⑱ double getDouble(String columnName)：以 Java 双精度形式获取当前行中指定名称列的值。

⑲ float getFloat(int columnIndex)：以 Java 浮点形式获取当前行中某序号列的值。

⑳ float getFloat(String columnName)：以 Java 浮点形式获取当前行中指定名称列的值。

㉑ int getInt (int columnIndex)：以 Java 编程语言中的整数形式获取此 ResultSet 对象当前行中指定序号列的值。

㉒ int getInt(String columnName)：以 Java 编程语言中的整数形式获取此 ResultSet 对象的当前行中指定名称列的值。

㉓ long getLong(int columnIndex)：以 Java 长整型形式获取当前行中指定序号列的值。

㉔ long getLong(String columnName)：以 Java 长整型形式获取当前行中指定名称列的值。

㉕ ResultSetMetaData getMetaData()：检索此 ResultSet 对象的列的数目、类型和属性。

㉖ Object getObject(int columnIndex)：以 Java 对象形式获取当前行中指定序号列的值。

㉗ Object getObject(String columnName)：以 Java 对象形式获取当前行中指定名称列的值。

㉘ int getRow()JDBC 2.0：检索当前行号。

㉙ short getShort(int columnIndex)：以 Java 编程语言中的 short 形式获取此 ResultSet 对象当前行中指定序号列的值。

㉚ short getShort(String columnName)：以 Java 编程语言中的 short 形式获取此 ResultSet 对象当前行中指定名称列的值。

㉛ Statement getStatement()JDBC 2.0：返回产生此 ResultSet 对象的"语句"。

㉜ String getString(int columnIndex)：以 Java 编程语言中的 String 形式获取此 ResultSet 对象当前行中指定序号列的值。

㉝ String getString(String columnName)：以 Java 编程语言中的 String 形式获取此 ResultSet 对象当前行中指定名称列的值。

㉞ Time getTime(int columnIndex)：以 Java 编程语言中的 java. sql. Time 对象形式获取此 ResultSet 对象的当前行中指定序号列的值。

㉟ Time getTime(String columnName)：以 Java 编程语言中的 java. sql. Date 对象形式获取此 ResultSet 对象的当前行中指定名称列的值。

㊱ Timestamp getTimestamp(String columnName)：以 Java 编程语言中的 java. sql. Timestamp 对象形式获取此 ResultSet 对象的当前行中指定名称列的值。

㊲ Timestamp getTimestamp(int columnIndex)：以 Java 编程语言中的 java. sql. Timestamp 对象形式获取此 ResultSet 对象的当前行中指定序号列的值。

(2) 与游标相关的方法

① boolean isAfterLast()：指示游标是否在结果集中的最后一行后面。

② boolean isBeforeFirst()：指示游标是否在结果集中的第一行前面。

③ boolean isFirst()：指示游标是否在结果集中的第一行上。

④ boolean isLast()：指示游标是否在结果集中的最后一行上。对于具有类型 TYPE_FORWARD_ONLY 的结果集，不支持此方法。

⑤ boolean last()：将游标移至结果集中的最后一行。

⑥ boolean next()：将游标从当前位置向下移动一行。

⑦ boolean previous()：将游标移至结果集中的前一行。

⑧ boolean relative(int rows)：将游标移动相对数量的行数，可以是正数或负数。

⑨ boolean absolute(int row)：将游标移至结果集中的给定行号。

⑩ void afterLast()：将游标移至结果集的末尾，正好在最后一行的后面。

⑪ void beforeFirst()：将游标移至结果集的前方，正好在第一行的前面。

⑫ boolean first()：将游标移至结果集中的第一行。

(3) 其他方法

① boolean wasNull()：报告读取的最后一列是否具有值 SQL NULL。

② int getType()JDBC 2.0：返回此结果集的类型。

③ SQLWarning getWarnings()：返回此 ResultSet 上的调用报告的首次警告。

④ void clearWarnings()：清除此 ResultSet 对象上报告的所有警告。

⑤ void close()：立即释放此 ResultSet 对象的数据库和 JDBC 资源，而不是等待对象自动关闭时才释放它们。

⑥ int findColumn(String columnName)：将给定 ResultSet 列名映射至其 ResultSet 列索引。

第七步 关闭创建的各个对象

操作完成以后，要把所有使用的 JDBC 对象全都关闭，以释放 JDBC 资源。关闭分别使用各自的 close()方法，关闭的顺序和声明顺序相反，先关闭结果集对象(如果有)，再关闭语句对象，最后关闭连接对象。

下面给出课堂练习 4-12 的 jsp 代码部分的示例，说明 JSP 通过 JDBC-ODBC 桥进行数据库操作的步骤。

示例 4-19 JSP 通过 JDBC-ODBC 桥进行数据库操作

```
<%@page contentType="text/html; charset=gb18030" language="java"%>
<%@page import="java.sql.*"%>
<HTML>                                    引入java.sql包
    <HEAD>
        <TITLE>使用 JDBC-ODBC 桥访问 Access数据库 </TITLE>
    </HEAD>
      <BODY>
    <%
        out.println("<CENTER>");
        out.println("<FONT size=4>学生信息</FONT>");
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                                                      载入驱动程序类别

        String url="jdbc:odbc:Student";
        String user="";                                 设置JDBC URL
        String pwd="";
        Connection con=DriverManager.getConnection(url,user,pwd);
                                                           获取数据库连接
```

```
        PreparedStatement pstmt=con.prepareStatement("SELECT * FROM student
                WHERE school =?", ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY);
                                                    预编译SQL语句
        pstmt.setString(1, "软件学院");    传递IN参数

        ResultSet rs=pstmt.executeQuery();
    %>                                           执行SQL语句，获得结果集
        <TABLE border=1>
        <TR>
        <TD align=center><B>姓名</B></TD>
        <TD align=center><B>学院</B></TD>
        <TD align=center><B>年龄</B></TD>
        </TR>
        <%
            rs.beforeFirst();    将游标移至结果集第一行之前

            while(rs.next())    将游标从当前位置向下移动一行
            {
    %>
        <TR ALIGN=CENTER>                              获取当前行中name列的值
        <TD><%=rs.getString("name")%></TD>
        <TD><%=rs.getString("school")%></TD>    获取当前行中school列的值
        <TD><%=rs.getInt("age")%></TD>
        </TR>                                          获取当前行中age列的值
        <%    }
            rs.close();          关闭结果集对象
            pstmt.close();       关闭语句对象
            con.close();         关闭连接对象
          %>
        </TABLE>
    </CENTER>
  </BODY>
<HTML>
```

向数据表 student 中添加三条记录，如图 4-24 所示。

student

id	name	school	age
1	张一	软件学院	20
2	李铭	财经学院	19
3	王圳	外语学院	21

图 4-24　student 数据表中的记录

示例 4-19 中的 search.jsp 执行的结果是把软件学院的学生显示在浏览器中，如图 4-25 所示。

为了便于读者理解，课堂练习 4-12 演示了如何使用 JDBC-ODBC 桥的方式进行数据库的一般性操作。示例源文件中所采用的数据库的连接方式，是对每一次数据库操作请求都要建立一个数据库的连接，对资源消耗很大，所以从第 5 章开始，数据库的连接技术

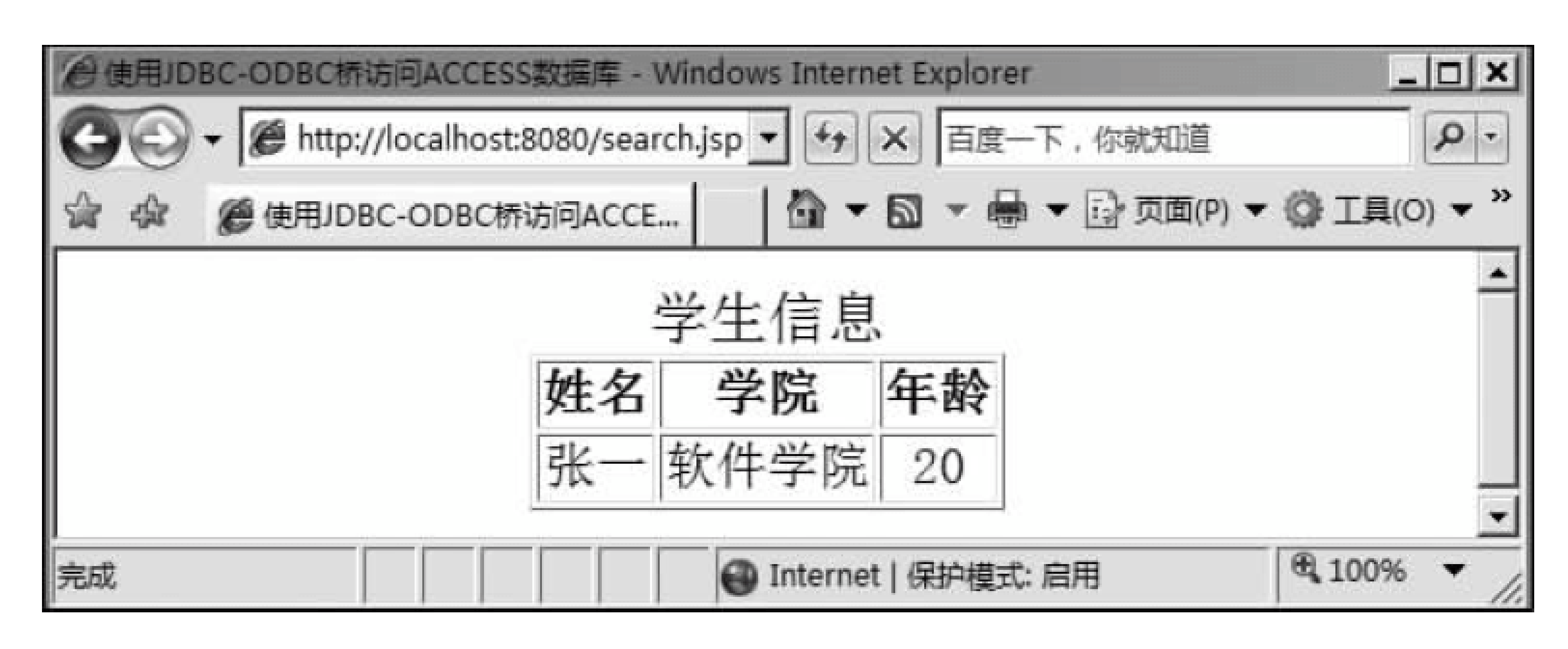


图 4-25 student 数据表中的记录

将采用第 3 章任务六中的连接池方式。

4.6.7 JDBC URL

示例 4-19 中连接数据库时使用了 JDBC URL。JDBC URL 是一种标识数据库的方法，可以使相应的驱动程序能够识别该数据库并与它建立连接。标准的 JDBC URL 的格式如下：

```
jdbc : <子协议名>: <子名称>
```

JDBC URL 由三个部分组成，各个部分之间用冒号分隔。＜子协议＞是指数据库连接的方式。＜子名称＞可以根据子协议的改变而变化。

JDBC-ODBC 桥驱动程序使用 ODBC 子协议。该子协议的 URL 格式如下：

```
jdbc:odbc:<data_source_name>[<attribute_name1>=<attribute_value1>]...
[<attribute_namen>=<attribute_valuen>]
```

下面列出常用数据库的 JDBC URL 的格式。

(1) Microsoft SQL Server 2000

驱动程序包名：msbase.jar mssqlserver.jar msutil.jar

驱动程序类名：com.microsoft.jdbc.sqlserver.SQLServerDriver

JDBC URL：jdbc:microsoft:sqlserver://＜server_name＞:＜port＞

默认端口为 1433。如果服务器使用默认端口，则 port 可以省略。

(2) Microsoft SQL Server 2005 JDBC Driver

驱动程序包名：sqljdbc.jar

驱动程序类名：com.microsoft.sqlserver.jdbc.SQLServerDriver

JDBC URL：jdbc:sqlserver://＜server_name＞:＜port＞

默认端口为 1433。如果服务器使用默认端口，则 port 可以省略。

(3) Oracle Thin JDBC Driver

驱动程序包名：ojdbc14.jar

驱动程序类名：oracle.jdbc.driver.OracleDriver

JDBC URL：jdbc:oracle:thin:@//＜host＞:＜port＞/ServiceName

或 jdbc:oracle:thin:@＜host＞:＜port＞:＜SID＞

(4) IBM DB2 Universal Driver Type 4

驱动程序包名：db2jcc.jar db2jcc_license_cu.jar

驱动程序类名：com.ibm.db2.jcc.DB2Driver

JDBC URL：jdbc:db2://＜host＞[:＜port＞]/＜database_name＞

(5) IBM DB2 Universal Driver Type 2

驱动程序包名：db2jcc.jar db2jcc_license_cu.jar

驱动程序类名：com.ibm.db2.jcc.DB2Driver

JDBC URL：jdbc:db2:＜database_name＞

(6) MySQL Connector/J Driver

驱动程序包名：mysql-connector-java-x.x.xx-bin.jar

驱动程序类名：com.mysql.jdbc.Driver

JDBC URL：jdbc:mysql://＜host＞:＜port＞/＜database_name＞

默认端口为 3306。如果服务器使用默认端口，则 port 可以省略。

(7) Informix JDBC Driver

驱动程序包名：ifxjdbc.jar

驱动程序类名：com.informix.jdbc.IfxDriver

JDBC URL：jdbc:informix-sqli://{＜ip-address＞|＜host-name＞}:＜port-number＞[/＜dbname＞]:INFORMIXSERVER=＜server-name＞

(8) Sybase Adaptive Server Enterprise JDBC Driver

驱动程序包名：jconn2.jar 或 jconn3.jar

驱动程序类名：com.sybase.jdbc2.jdbc.SybDriver 或 com.sybase.jdbc3.jdbc.SybDriver

JDBC URL：jdbc:sybase:Tds:＜host＞:＜port＞

默认端口为 5000。

(9) Sybase Adaptive Server Anywhere or Sybase IQ JDBC Driver

驱动程序包名：jconn2.jar 或 jconn3.jar

驱动程序类名：com.sybase.jdbc2.jdbc.SybDriver 或 com.sybase.jdbc3.jdbc.SybDriver

JDBC URL：jdbc:sybase:Tds:＜host＞:＜port＞?ServiceName=＜database_name＞

默认端口为 2638。

课后练习

1. 仿照新浪会员中心注册页，利用 HTML 表单编写一个注册页。

2. 如何使课堂练习 3-2 输出的表格中的相邻行的底色不同？试实现之。

提示：使用数组元素表示颜色。

3. JSP 的指令标签有哪些？它们的语法格式是怎样的？分别起到什么作用？

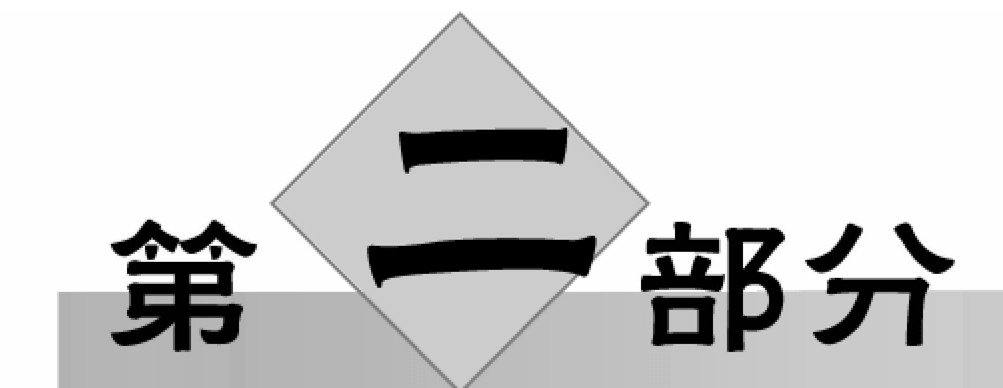
4. JSP 的动作标签有哪些？它们的语法格式是怎样的？分别起到什么作用？

5. session 对象的工作原理是什么？

6. session 对象和 application 对象有什么区别？

7. 试将课堂练习 4-12 中的 Access 数据库替换为 MySQL 数据库，查询年龄大于 20 岁的学生，并将其基本信息显示在浏览器中。





系统数据访问功能模块的设计开发



第 5 章　商品信息的显示和查询

本章学习要点：

- 熟练掌握 JavaBean 的基本要素；
- 熟练掌握实体类的编写规范；
- 熟练掌握 DAO 类中各种查询方法的编写技巧；
- 熟练掌握 SQL SELECT 语句的各种用法；
- 熟练掌握<jsp:useBean><jsp:setProperty><jsp:getProperty>动作标签的用法；
- 了解 JavaBean 的作用范围。

5.1　任务一：商品展示的实现

问题：使用 JSP+JavaBean 技术，如何将数据表中的信息显示到浏览器中呢？使用 JavaBean 又有哪些 Java Web 程序员普遍接受的约定需要遵守呢？

任务目标：

将第 3 章中创建的 goods 表中的记录采用 JSP+JavaBean 的技术显示在浏览器中。

技能训练：

- 实体类的编写规范。
- DAO 类 findAll()方法的编写规范。
- JSP 与 JavaBean 相关的动作标签的使用。

本任务中提出使用 JSP+JavaBean 技术来完成这一任务，那么什么是 JavaBean 呢？

5.1.1　JavaBean 的定义

JavaBean 一种满足特殊基本要素的 Java 类，这些要素使它们可以很容易地被重复使用并且可以进行组合，从而可以应用在不同的应用程序中。

JavaBean 的基本要素如下：

(1) JavaBean 必须放在一个包(package)中；

(2) JavaBean 必须有一个不带参数的构造器；

(3) Bean 类应该没有任何公共变量,Bean 类的成员变量都是 private 类型的;

(4) 通过 public 的 getters 和 setters 方法来读取或者设置成员变量的值,getters 和 setters 方法和对应的实例变量之间的命名也要遵循一定的约定:成员变量名称首字母必须小写,而对应的 getter/setter 方法采用"骆驼命名法"(CamelCase)命名,即:set/get+将变量首字母改成大写。例如,有一个实例变量 age,它的 setter 方法名称应该定义成 setAge(),而它的 getter 方法名称应该定义成 getAge()。这种命名规则有一个特例,如果成员变量是 boolean 类型的,那么,它对应的 getter 方法应该采用"is"开头,例如,假设有一个实例变量 married 表示"婚否"(true 或者 false),那么,它对应的 getter 方法名称应该是 isMarried()。而不是 getMarried(),而对于 setter 方法,规则不变。

5.1.2 商品信息实体 Bean 的编写

最常用的是用实体 Bean 代表关系库中的数据,一个简单的实体 Bean 可以定义成代表数据库表的一个记录,其名字一般与所对应的数据表同名,并且首字母大写,其成员变量(也称为属性)的名字和类型与数据表中的列的名字和类型相同。实体类一般被创建后放在 entity 包中。下面分两步来完成第一个 JavaBean 的编写。

第一步 如图 5-1 所示为 goods 表创建一个与之相对应的实体 Bean,并根据数据表完成属性的编写。

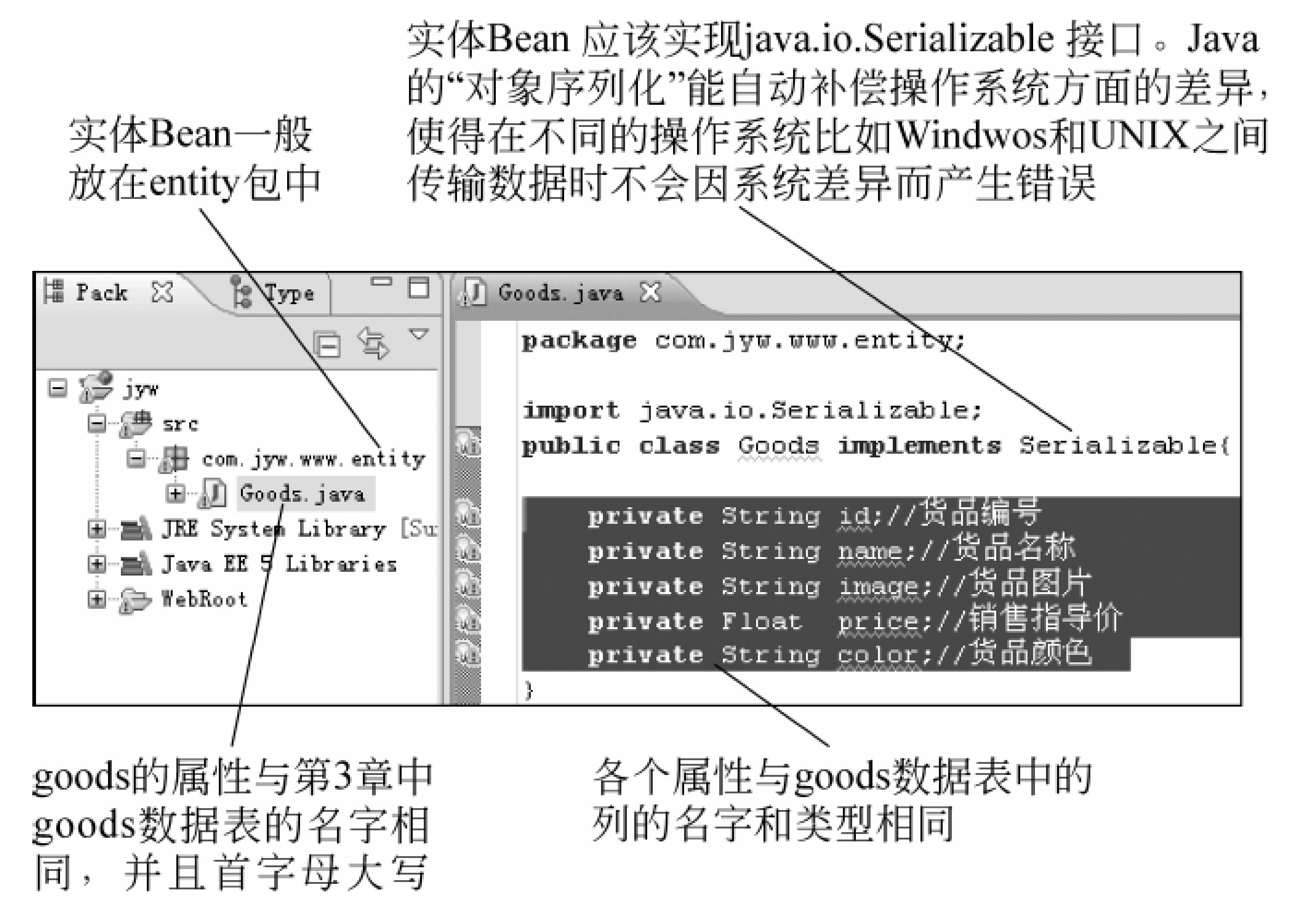


图 5-1 实体 bean 属性的编写

第二步 属性写好后,可以利用 MyEclispe 中 source→Generate Getters and Setters 命令来自动生成 getters 和 setters 方法,如图 5-2 所示。

在图 5-3 所示的对话框中勾选需要创建 setters 和 getters 的属性。

完成后就会生成如下所示的一个实体 Bean。

示例 5-1 Goods.java

```
package com.jyw.www.entity;
```

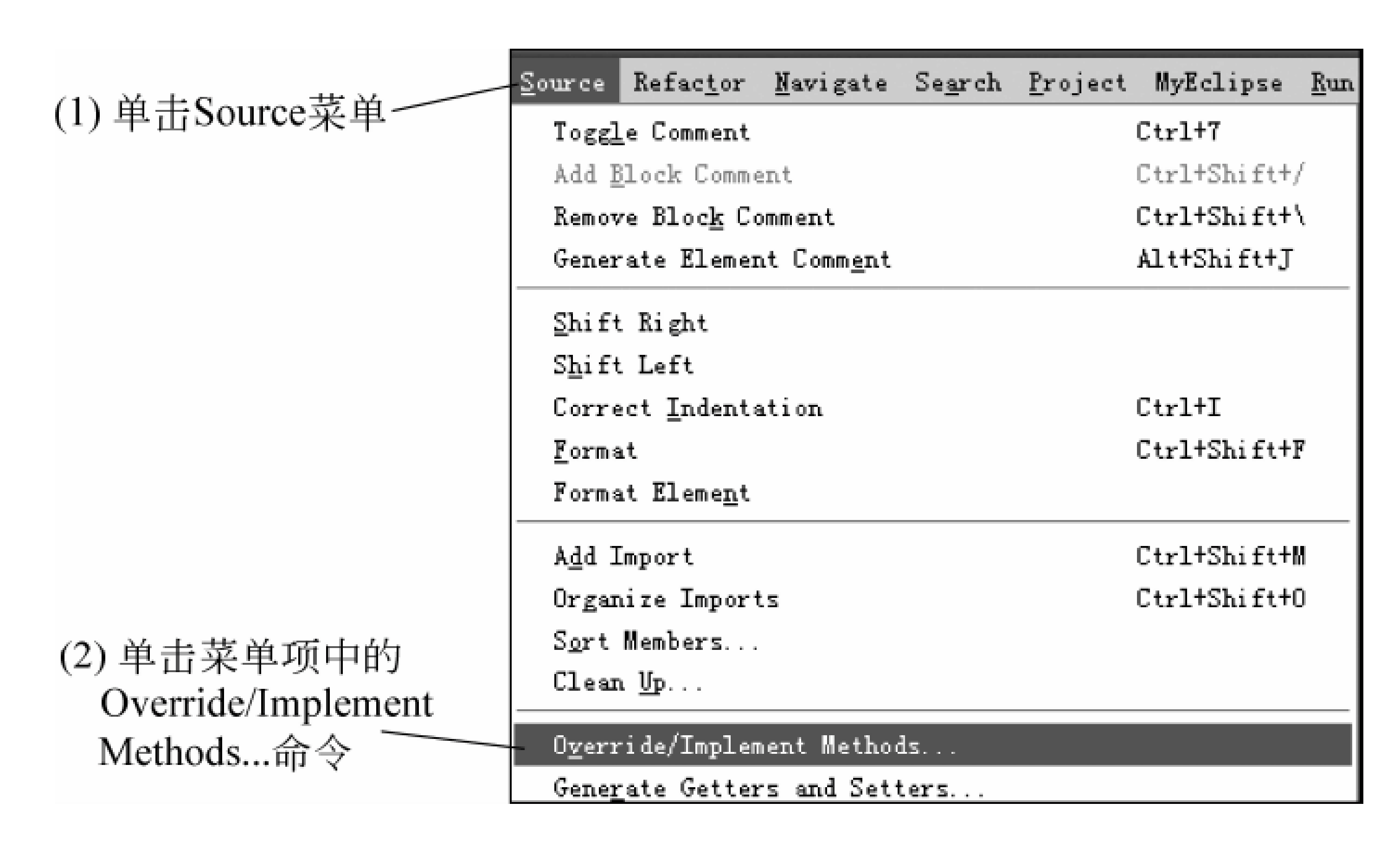


图 5-2 自动生成 setters 和 getters 方法的菜单选项

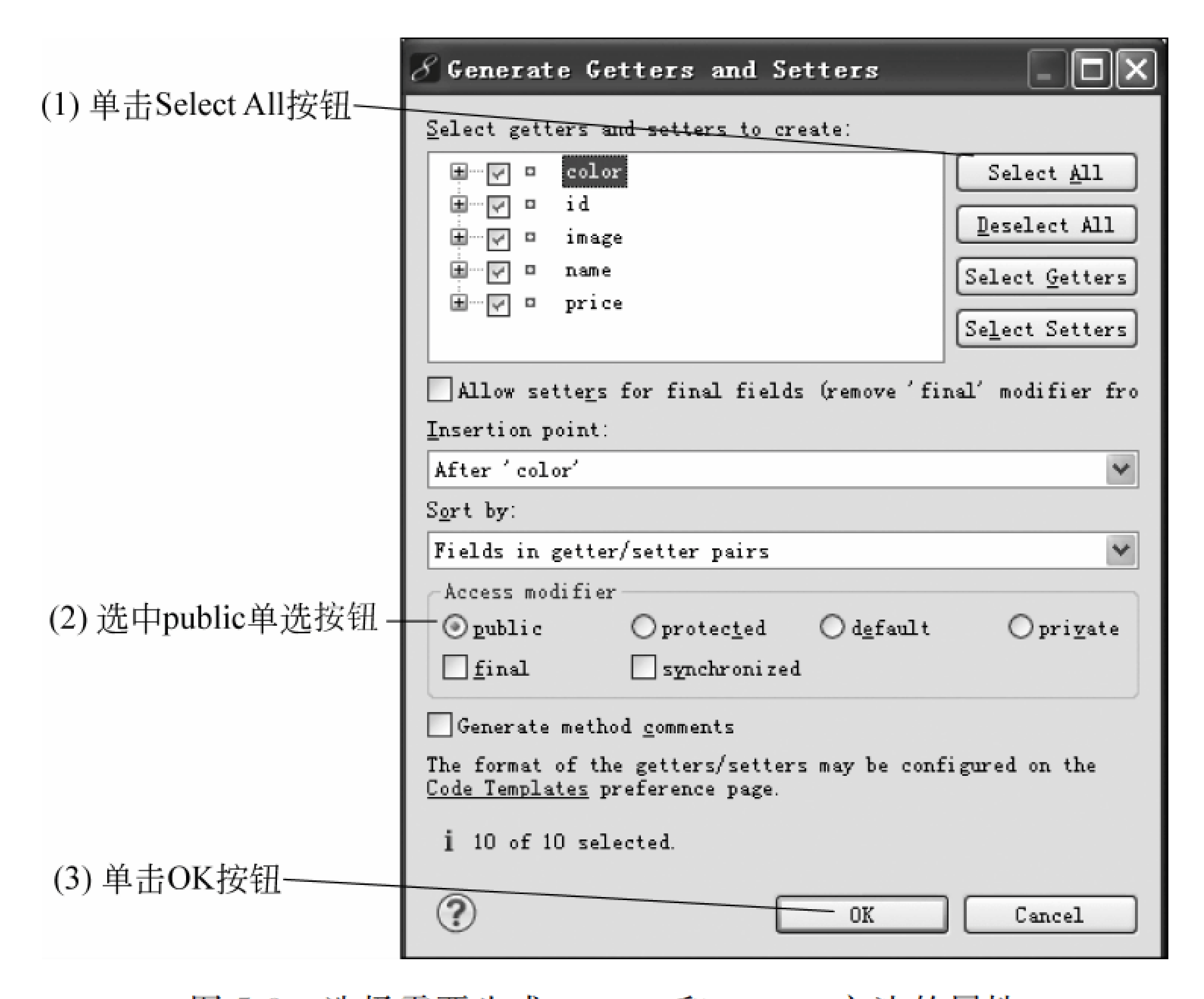


图 5-3 选择需要生成 getters 和 setters 方法的属性

```
import java.io.Serializable;
public class Goods implements Serializable{
    private String id;                   //货品编号
    private String name;                 //货品名称
    private String image;                //货品图片
    private Float  price;                //销售指导价
    private String color;                //货品颜色
    private String size;                 //货品尺码
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id=id;
```

```
        }
        public String getName() {
            return name;
        }
        public void setName(String name) {
            this.name=name;
        }
        public String getImage() {
            return image;
        }
        public void setImage(String image) {
            this.image=image;
        }
        public Float getPrice() {
            return price;
        }
        public void setPrice(Float price) {
            this.price=price;
        }
        public String getColor() {
            return color;
        }
        public void setColor(String color) {
            this.color=color;
        }
        public void setSize(String size) {
            this.size=size;
        }
        public String getSize() {
            return size;
        }
}
```

对照JavaBean的基本要素，体会getters和setters方法的编写规则

5.1.3 DAO 类中 findAll 方法的编写

本任务的目标是从数据表中取出数据并显示在浏览器中，所以编写与数据库操作相关的 DAO :(Data Access Objects)类。DAO 的成员方法主要完成对数据表中的数据做增、删、改、查等操作。DAO 类一般被创建在 dao 包中，其命名约定是“所要操作的数据表的名字+DAO”，如图 5-4 所示。

DAO 中的各方法的命名也有约定，实现查询操作的方法一般以 find 开头，比如读取所有记录的方法命名为 findAll()。以下示例是 GoodsDAO 方法中 findAll()方法的源代码。

示例 5-2 GoodsDAO 类中的 findAll()方法

```
public ArrayList<Goods> findAll() {
```



图 5-4　创建 GoodsDAO 类

```
    ArrayList<Goods>ret=new ArrayList<Goods>();
    try {
        InitialContext ctx=new InitialContext();
        DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
        Connection conn=ds.getConnection();

        Statement stm=conn.createStatement();
        ResultSet res=stm.executeQuery("SELECT * FROM goods");
        while (res.next()) {
            Goods goods=new Goods();
            goods.setId(res.getString("id"));
            goods.setColor(res.getString("color"));
            goods.setName(res.getString("name"));
            goods.setImage(res.getString("image"));
            goods.setPrice(res.getFloat("price"));
            goods.setSize(res.getString("size"));
            ret.add(goods);
        }
        res.close();
        stm.close();
        conn.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return ret;
}
```

从连接池中获取数据库连接

执行对goods表的查询语句

将goods表中的记录从结果集中逐条取出并放在ArrayList中

将数据库连接放回池中

将从数据表中读出的记录信息返回给调用者

5.1.4　编写显示商品信息的 JSP 文件

实体类和 DAO 类写好后，还要编写 JSP 文件来显示从数据表中读取的记录。JSP 文件创建在项目的 WebRoot 目录下，如图 5-5 所示。

在接着出现的对话框中按照图 5-6 的指示操作。

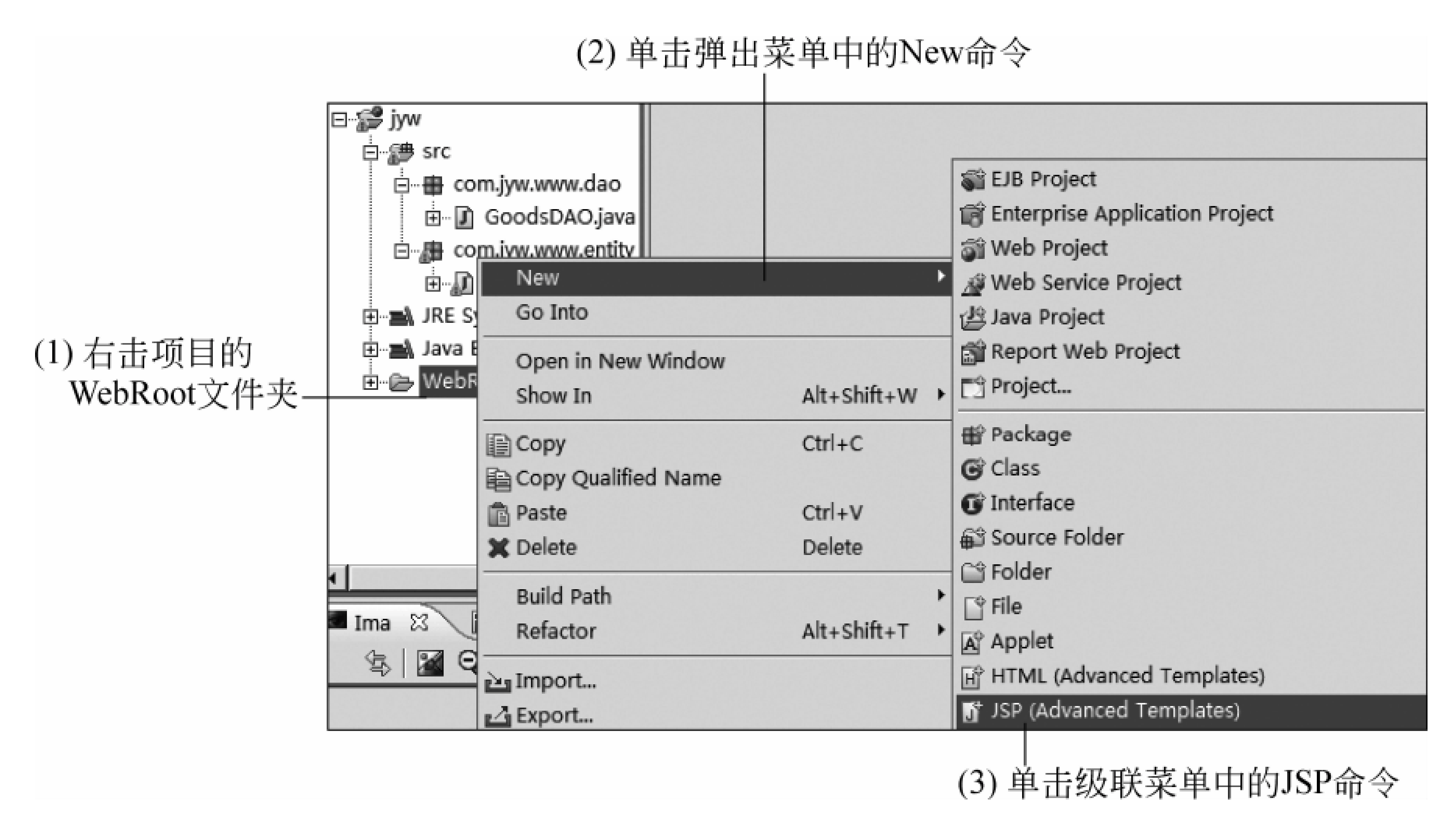


图 5-5　在项目中创建 JSP 文件

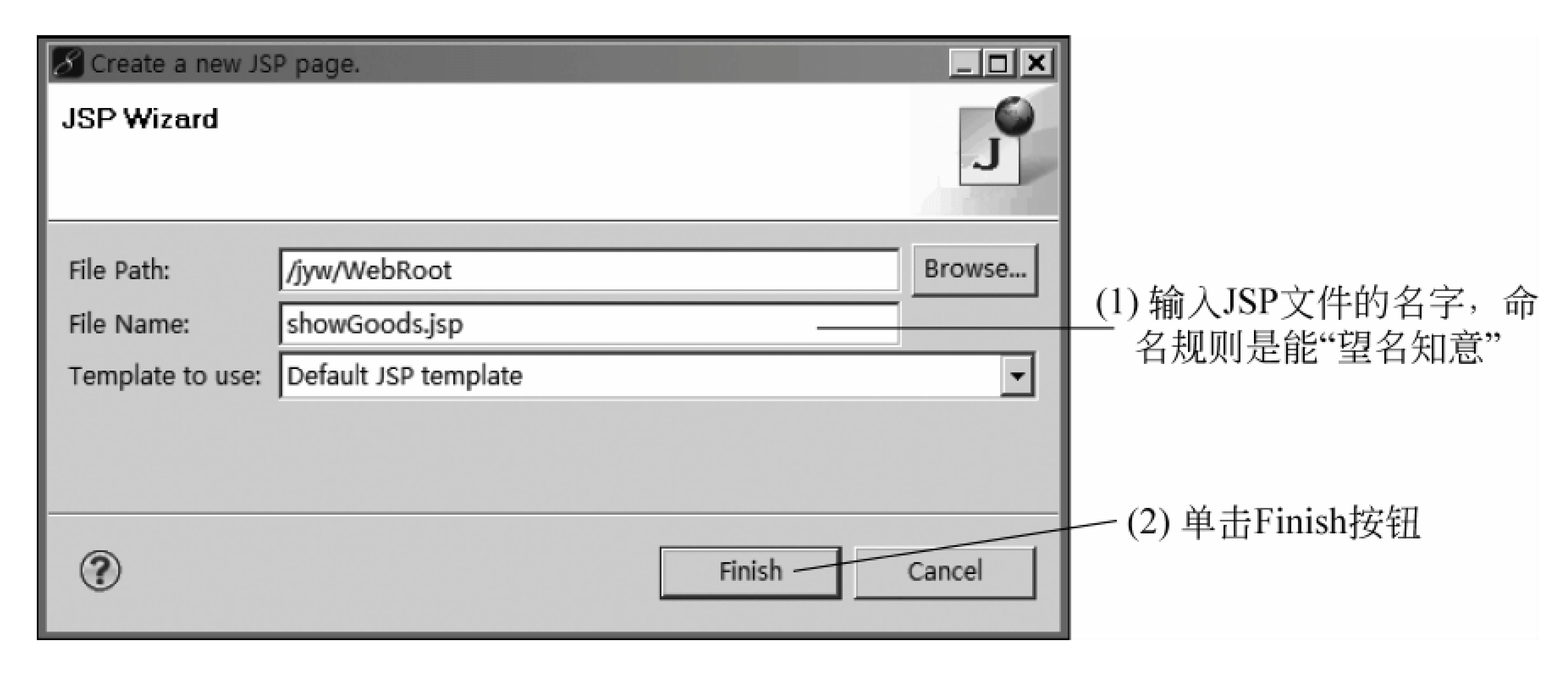


图 5-6　为 JSP 文件命名

示例 5-3 给出了 showGoods.jsp 的源代码，对 goods 表中所有的商品基本信息的记录每 4 个为一行进行显示。

示例 5-3　showGoods.jsp

```
<%@page contentType="text/html; charset=gb18030" language="java"%>
<%@page import="com.jyw.www.entity.*,com.jyw.www.dao.*,java.util.*"%>
<HTML>
<BODY>
<CENTER>
商品展示
<TABLE border=1>
<%
    GoodsDAO goodsDao=new GoodsDAO();
    List<Goods>goodsList=goodsDao.findAll();
    Iterator it=goodsList.iterator();
```

Page指令定义了编码集，需要引入包等信息

调用GoodsDAO中的findAll()方法，获得goods表中所有的记录信息

```
    while (it.hasNext()) {
%>
    <TR ALIGN=CENTER>
<%
    for (int j=0; j<4; j++) {
     Goods goods=(Goods) it.next();
%>
  <TD>
    <TABLE border=1>
        <TR><TD colspan="3">

    <img src="/images/<%=goods.getImage()%>" height=140  width=120>
    </TD></TR>
    <TR ALIGN=CENTER><TD colspan="3"><%=goods.getId()%></TD></TR>
    <TR ALIGN=CENTER><TD colspan="3"><%=goods.getName()%></TD></TR>
    <TR><TD><%=goods.getColor()%></TD>
        <TD><%=goods.getSize()%></TD>
        <TD>￥<%=goods.getPrice()%></TD>
      </TR>
     </TABLE>
    </TD>
<%
    if (!it.hasNext())
                    break;
    }
%>
    </TR>
<%
    }
%>
</TABLE>
</BODY>
</CENTER>
</HTML>
```

使用双重循环实现商品信息每4条在一行的显示

图片放在根目录的images子目录下

文件编写完成后，就可以按照第 2 章的方法启动 Web 服务器，将项目发布到服务器上，再在浏览器 URL 中输入正确的访问地址，就可以看到如图 5-7 的显示效果了。



图 5-7　商品信息显示效果

5.2 任务二：商品库存信息的排序显示

问题：在现实生活中，为了方便检索，常常会对数据的集合进行排序，比如“花名册”是按照同学们的学号进行了排序。那么如果希望对任务一中的结果进行排序显示，使用 JavaBean 技术如何实现呢？

任务目标：

将第 3 章中创建的 stock 表中的记录采用 JSP＋JavaBean 的技术显示在浏览器中，并按照分店号进行排序。

技能训练：

- 完善实体类的编写规范。
- 完善 DAO 类 findAll()方法的编写规范。
- JSP 与 JavaBean 相关的动作标签的用法。
- 排序查询。

任务二要求按照分店号及 stock 表中的 storenum 列进行排序，在完成任务一后，得到了所有商品基本信息的列表，在没有指明按照某个列排序的情况下，MySQL Select 默认排序方法是按照物理存储顺序显示。

5.2.1 对视图的排序查询

首先要在项目的 entity 包中添加一个与 stock 表相对应的实体 Bean Stock，请读者自行完成。

然后在 DAO 包中添加 StockDAO 类，其中的 findAll()方法可以完成排序查询，需要用 SELECT 语句的 ORDER BY 子句来指明按照哪个列进行排序，其语法规则是：

```
SELECT  FROM 数据表的名字 ORDER BY 列的名字
```

示例 5-4 StockDAO 类中的 findAll()方法

```
public List<Stock>findAll() {
    String selectsql="SELECT * FROM stock ORDER BY storenum ";
    List<Stock>ret=new ArrayList<Stock>();
    try {
      InitialContext ctx=new InitialContext();
      DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
      Connection conn=ds.getConnection();
      Statement stm=conn.createStatement();
      ResultSet res=stm.executeQuery(selectsql);
```

按照分店号即storenum列排序

```
        while (res.next()) {
          Stock stock=new Stock();
          stock.setId(res.getInt("id"));
          stock.setGoodsid(res.getString("goodsid"));
          stock.setSize(res.getString("size"));
          stock.setStock(res.getInt("stock"));
          stock.setStorenum(res.getInt("storenum"));
          ret.add(stock);
        }
        res.close();
        stm.close();
        conn.close();
      } catch (Exception e) {
        e.printStackTrace();
      }
      return ret;
}
```

while循环将stock表中的记录从结果集中逐条取出并放在ArrayList中，再返回给调用方

5.2.2　在 JSP 中使用增强的 FOR 循环

Jdk1.5 增加了增强的 for 循环，使得遍历数组或集合时更加简便。其语法规则如下：

遍历集合的语法为：

```
for (Type value : Iterable) {
    expression value;
}
```

遍历数组语法为：

```
for (Type value : array) {
    expression value;
}
```

示例 5-5 的 showStock.jsp 中使用增强的 for 循环显示 Java ArrayList 中的信息。

示例 5-5　showStock.jsp

```
<%@page contentType="text/html; charset=gb18030" language="java"%>
<%@page import="com.jyw.www.entity.*,com.jyw.www.dao.*,java.util.*"%>
    <HTML>
    <HEAD><TITLE>利用 JDBC-ODBC 桥访问 ACCESS 数据库</TITLE></HEAD>
    <BODY>
    <TABLE border=1><TR><TD align=center><B>序号</B></TD>
                <TD align=center><B>商品编号</B></TD>
                <TD align=center><B>尺码</B></TD>
                <TD align=center><B>库存量</B></TD>
                <TD align=center><B>分店编号</B></TD>
              </TR>
          <%
                StockDAO stockdao=new StockDAO();
```

```
                List<Stock> stocklist=stockdao.findAll();
                for (Stock stock: stocklist) {
            %>
            <TR ALIGN=CENTER>
            <TD><%=stock.getId()%></TD>
                <TD><%=stock.getGoodsid()%></TD>
                <TD><%=stock.getSize()%></TD>
                <TD><%=stock.getStock()%></TD>
                <TD><%=stock.getStorenum()%></TD>
            </TR>
<%     }     %>
        </TABLE>
    </BODY>
</HTML>
```

增强的for循环，对集合进行遍历

如图 5-8 所示为 showStock.jsp 执行的结果。

从运行结果中可以看到，列表按照分店编号从低到高进行了排序。如果希望列表按照分店号从高到低进行排序，也就是“反序”，就需要在 ORDER BY 子句后面加关键字 DESC，如下所示。

地址(D) http://localhost:8080/showStock.jsp

库存商品列表

序号	商品编号	尺码	库存量	分店编号
1	JYW12CKB0	M	40	0
2	JYW12CKN1	L	50	0
3	JYW12CKY2	S	100	0
4	JYW12CKB0	S	20	0
5	JYW12CKN1	M	3	1

图 5-8 库存商品信息的显示效果

```
String selectsql=" SELECT * FROM stock ORDER BY storenum DESC"
```

DESC表明反序显示查询结果

5.3 任务三：查询各分店的库存商品详细信息

问题：我们在购物的时候常常会遇到这样的情况：想要购买某款服装或者鞋子，但摆放在货架上的商品并不适合我们想要的尺码，这时，店员会去查询店里是否还有适合我们想要的尺码的库存。这个功能采用 JSP＋JavaBean 技术是如何实现的呢？

任务目标：

查询用户指定店的库存商品，包括图片、指导价格等的详细信息。

技能训练：

- 视图对应的实体类的编写规范。
- DAO 类 findByXxxx()方法的编写规范。

完成了第 3 章的任务四后，获得了由基表 goods 和 stock 生成的视图 stockinfo，这个视图中汇总了任务三要求查询的所有信息，视图可以看做是“虚拟的表”，Java 对数据库中视图的访问，可以使用访问数据表的技术来完成。

与编写对数据表访问的代码相似，首先编写与 stockinfo 视图对应的实体类，请读者参照前面的小节自行完成，然后编写 StockInfoDAO 类中的方法。这里需要根据用户指定的店来查询该店的库存，这是一个条件查询。

5.3.1　编写 DAO 类中的 findByXxxx　方法

条件查询方法的命名约定为 findByXxxx(Xxxx xxxx)，例如按照分店名进行查找的方法命名为 findByStoreName (String storename)。示例 5-6 为 StockInfoDAO 类中的 findByStoreName()方法的代码。

示例 5-6　StockInfoDAO 中用于按分店编号查询的 findByStoreName()方法

```
public List<StockInfo>findByStoreName(String storename) {
    List<StockInfo>ret=new ArrayList<StockInfo>();
    try{
        InitialContext ctx=new InitialContext();
        DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
        Connection conn=ds.getConnection();
        PreparedStatement pstmt=conn.prepareStatement("SELECT * FROM stockinfo
                                WHERE storename=?");
        storename=new String(storename.getBytes("iso-8859-1"),"GB18030");
        pstmt.setString(1, storename);
        ResultSet res=pstmt.executeQuery();
        while (res.next()) {
            StockInfo goods=new StockInfo();
            goods.setId(res.getString("id"));
            goods.setName(res.getString("name"));
            goods.setSize(res.getString("size"));
            goods.setColor(res.getString("color"));
            goods.setImage(res.getString("image"));
            goods.setPrice(res.getFloat("price"));
            goods.setStock(res.getInt("stock"));
            goods.setStorename(storename);
            ret.add(goods);
        }
        res.close();
        pstmt.close();
        conn.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
    return ret;
}
```

将用户指定的分店名作为入口参数

预编译带条件的SELECT语句

5.3.2　编写与用户查询相关的 JSP 文件

下面编写 showStockInfo.jsp 文件完成用户选择分店名和根据用户的选择显示查询结果的功能。

示例 5-7　showStockInfo.jsp 中用于查询的表单部分如下

```
<%@page contentType="text/html; charset=gb18030" language="java"%>
```

```
<%@page import="com.jyw.www.entity.*,com.jyw.www.dao.*,java.util.*"%>
<HTML>
    <BODY><CENTER>
    库存商品列表
    <form action="searchStock.jsp" method="post">
    店名<select name="storenum">
         <option></option>
         <option value="1">佳衣屋海岸城店</option>
        </select>
      <input name="submit" type="submit" value="查询库存">
    </form>
```

表单部分在 IE 浏览器中的显示结果如图 5-9 所示。



图 5-9　用户选择分店的列表菜单

示例 5-8　showStockInfo.jsp 中显示查询结果的部分如下：

```
<TABLE border=1>
<%
        List<StockInfo>goodsList=stockinfoDAO.findByStoreName
                         (request.getParameter("storename"));
        Iterator it=goodsList.iterator();
        while (it.hasNext()) {
%>
      <TR ALIGN=CENTER>
<%
          for (int j=0; j<4; j++) {
              StockInfo goods=(StockInfo) it.next();
%>
      <TD>
        <TABLE border=1>
        <TR><TD colspan="4">
        <img src="/images/<%=goods.getImage()%>" height=220  width=180>
      </TD></TR>
      <TR ALIGN=CENTER><TD colspan="4"><%=goods.getId()%></TD></TR>
      <TR ALIGN=CENTER><TD colspan="4"><%=goods.getName()%></TD></TR>
      <TR><TD><%=goods.getSize()%></TD>
          <TD><%=goods.getColor()%></TD>
          <TD>￥<%=goods.getPrice()%></TD>
          <TD><%=goods.getStock()%></TD>
      </TR>
```

以用户选择的分店名作为入口参数，调用示例5-6中的findByStoreName()方法

```
        </TABLE>
        </TD>
<%
        if (!it.hasNext())
                break;
        }
%>
        </TR>
<%
        }
%>
</TABLE>
```

查询“佳衣屋总店华强北店”的库存商品详细信息的结果如图 5-10 所示。



图 5-10　查询指定分店的库存商品信息

5.4　任务四：用多个条件查询库存商品信息

问题：我们在购物的时候还会遇到这样的情况：我们想要购买的某款服装或者鞋子的尺码在当前这个分店已经售完，这时店员常常会去查询其他分店是否还有我们想要的款式和尺码的商品，这个查询功能使用 JSP+JavaBean 技术是怎样实现的呢？

任务目标：

用户可以指定商品编号、名称、颜色、尺码、库存量中的一个或多个条件查询库存商品（包括图片、指导价格等）的详细信息，其中商品名称是模糊查询。

技能训练：

- 视图对应的实体类的编写规范。
- DAO 类 findByExample()方法的编写规范。

5.4.1 编写 DAO 类中的 findByExample 方法

任务四要求完成一个多条件的查询，在编写条件 SELECT 语句时首先应明确用户指定的条件是哪几个，将用户指定的条件写在 SELECT 语句的 WHERE 子句中，即可以构成一个完整的可执行的 SELECT 语句，并将查询结果返回。可以在 StockInfoDAO 中添加一个 findByExample()方法实现此功能，此方法需要一个实体 Bean 类型的入口参数。

示例 5-9 StockInfoDAO 中的 findByExample()方法

```
public List<StockInfo>findByExample(StockInfo goods) {
    String selectsql=" SELECT * FROM goods";
    Boolean isFirstCondition=true;
    if (goods.getId() !=null) {
        selectsql=selectsql+" WHERE "+" id='"+goods.getId()+"'";
        isFirstCondition=false;
    }
    String name=goods.getName();
    if (name !=null) {
      try {
        name=new String(name.getBytes("ISO-8859-1"));
      } catch (UnsupportedEncodingException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
      }
      if (isFirstCondition) {
        selectsql=selectsql+" WHERE "+" name LIKE '%"+name+"%'";
        isFirstCondition=false;
      } else
        selectsql=selectsql+" AND "+" name LIKE '%"+name+"%'";
    }
    if (goods.getSize() !=null) {
        if (isFirstCondition) {
            selectsql=selectsql+" WHERE "+" size='"+goods.getSize()+"'";
            isFirstCondition=false;
        } else
        selectsql=selectsql+" AND"+" size='"+goods.getSize()+"'";
    }
    String color=goods.getColor();
    if (color !=null) {
        try {
            color=new String(color.getBytes("ISO-8859-1"));
```

入口参数为StockInfo类型的Bean，Bean中值为null的属性表示不是查询条件

逐个判断入口参数中各个属性是否为null，不为null的属性要添加到WHERE子句中作为一个条件

如果是第一个条件，前面须加WHERE关键字

如果不是第一个条件，与前面条件间须加and关键字

要求针对货品名进行模糊查询

有中文参数的，要进行编码转换

```
    } catch (UnsupportedEncodingException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
if (isFirstCondition) {
    selectsql=selectsql+" WHERE "+" color='"+color+"'";
    isFirstCondition=false;
} else
    selectsql=selectsql+"AND "+" color='"+color+"'";
}
Integer stock=goods.getStock();
if (stock !=null) {
      switch (stock) {
```

对于列表菜单提交的参数，用switch case语句来处理

```
        case 1:
            if (isFirstCondition) {
                selectsql=selectsql+" WHERE "+" stock<=20";
                isFirstCondition=false;
            } else
                selectsql=selectsql+" AND "+" stock<=20";
              break;
        case 2:
            if (isFirstCondition) {
                selectsql=selectsql+" WHERE "+" stock>20 and stock<=50";
                isFirstCondition=false;
            } else
                selectsql=selectsql+" AND "+" stock>20 and stock<=50";
            break;
        case 3:
            if (isFirstCondition) {
                selectsql=selectsql+" WHERE "+" stock>50";
                isFirstCondition=false;
            } else
            selectsql=selectsql+" AND "+" stock>50";
            break;
        }
      }

List<StockInfo>ret=new ArrayList<StockInfo>();
    try {
        InitialContext ctx=new InitialContext();
        DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
        Connection conn=ds.getConnection();
        Statement stm=conn.createStatement();
        ResultSet res=stm.executeQuery(selectsql);
        while (res.next()) {
            StockInfo fGoods=new Goods();
            fGoods.setId(res.getString("id"));
            fGoods.setName(res.getString("name"));
            fGoods.setSize(res.getString("size"));
```

```
                fGoods.setColor(res.getString("color"));
                fGoods.setImage(res.getString("image"));
                fGoods.setPrice(res.getFloat("price"));
                fGoods.setStock(res.getInt("stock"));
                ret.add(fGoods);
            }
                res.close();
                stm.close();
                conn.close();
        } catch (Exception e) {
                e.printStackTrace();
        }
            return ret;
    }
```

5.4.2 编写与用户多条件查询相关的 JSP 文件

下面编写用户输入或者选择查询条件的 JSP 页面。

示例 5-10 searchStock.jsp 中用于查询的表单部分如下：

```
<%@page contentType="text/html; charset=gb18030" language="java"%>
    查询结果
    <form action="searchStock.jsp" method="post">
        商品编号<input name="id" type="text">
        商品名称<input name="name" type="text">
        颜色<input name="color" type="text" size="10">
        尺码<select name="size">
                <option></option>
                <option>S</option>
                <option>M</option>
                <option>L</option>
                <option>XL</option>
        </select>
    库存量<select name="stock">
                <option></option>
                <option value="1">20 以下</option>
                <option value="2">20-50</option>
                <option value="3">50 以上</option>
        </select>
    <input name="submit" type="submit" value="查询">
  </form>
```

注意：为了方便处理，应将库存量选项的值设置为整数

在 IE 浏览器中的显示结果如图 5-11 所示。

用户提交的条件就是查询条件，根据第 4 章中对 JSP 内置对象 request 的介绍可以知道，通过 request.getParameter()方法可以读取用户通过表单元素提交的参数，再调用实体 Bean 的 setters 方法并用用户提交的参数来设置 Bean 的相应属性。但是，<jsp:useBean>和<jsp:setProperty>动作标签提供了更加便捷的方法来完成这一操作。

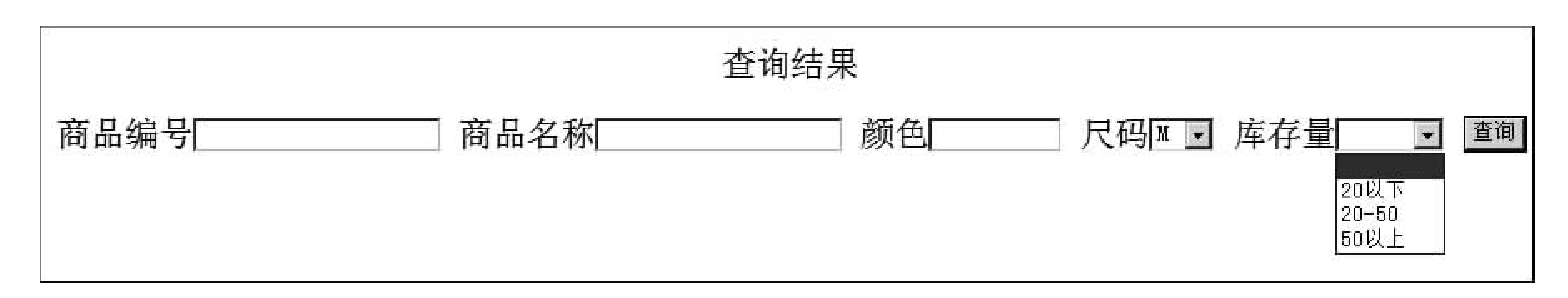


图 5-11　不确定多条件的用户查询界面

5.4.3　用 jsp:useBean 创建一个 Bean 实例

在 JSP 页面中可以使用<jsp:useBean>动作来创建一个 Bean 实例。<jsp:useBean>的语法规则如下。useBean 的定义方式有两种，分别如下面的两段代码所示。

```
<jsp:useBean
        id="beanInstanceName"
        scope="page | request | session | application"
        {
            class="package.class" |
            type="package.class" |
            class="package.class" type="package.class" |
            beanName="[package.class | <%=expression %>]" type="package.class"
        }
/>
```

和

```
<jsp:useBean
        id="beanInstanceName"
        scope="page | request | session | application"
        {
            class="package.class" |
            type="package.class" |
            class="package.class" type="package.class" |
            beanName="[package.class | <%=expression %>]" type="package.class"
        }>
<jsp:setProperty.../>
<jsp:useBean/>
```

<jsp:useBean>起始标记和结束标记之间的语句只有在新建 JavaBean 实例的时候才会执行，如果是使用已经存在的 JavaBean 实例，则不会执行。第二种方式可用于初始化一些属性。

注意：*<jsp:setProperty>可以单独放到外面，并独立于<jsp:useBean>而存在。*

下面是对 useBean 动作属性的简要说明。

- scope：用于指定 JavaBean 的作用范围，可能的取值为 page、request、session、application，它们的作用范围从小到大。
- id：定义 Bean 实例的名字。在 scope 定义的范围内，使用此名字来分辨不同的

Bean。这个变量名对大小写敏感。

- class：使用 new 关键字以及 class 构造器来创建指定类的 bean 实例。这个类不能是抽象的，必须有一个公用的、没有参数的构造器，package 的名字对大小写敏感。
- type：指定 JavaBean 对象的类型，通常用在当对应的 scope 中已经存在 JavaBean 对象的时候，这个时候使用 type 将不会产生新的对象。另外，它也可以和 class 属性结合使用，此时，type 指定的类可以和 class 指定的类是同一个类，或者是 class 指定类的父类、父接口。
- beanName：使用 java.beans.Beans.instantiate()方法，从一个类或者一个系列模板来实例化一个 JavaBean，并且指定 JavaBean 的类型为 type 属性指定的类型。这个属性通常并不使用。

注意：使用 jsp:useBean 可以得到一个 JavaBean 的实例，但是，这个实例不一定是新建的，如果已经存在一个与指定 id 同名、相同 scope 的 JavaBean，那么就不会新建一个 JavaBean 实例，而是直接使用已经存在的实例。

在 searchStock.jsp 中，添加下面语句来获得一个 StockInfo 类型的实体 Bean 实例 stockinfo。

```
<jsp:useBean id="stockinfo" scope="request"  class="com.jyw.www.entity.
StockInfo"/>
```

5.4.4 jsp:setProperty 关联查询参数与实体 Bean 的属性

如果请求参数和 JavaBean 的属性名称一致，可以使用下面的方式将所有的属性和相同名称的请求参数关联起来：只需要在 setProperty 中将“ * ”作为 property 的值：

```
<jsp:setProperty name=" stockinfo " property=" * "/>
```

通过这种方式，可以非常简便地将 JavaBean 的属性和请求参数关联起来。如果有部分的 JavaBean 属性无法找到对应的请求参数，则不会采取任何行动。

示例 5-11 searchStock.jsp 中关联表单参数与实体 Bean 属性的部分代码如下：




```
<%@page import="java.util.*,com.jyw.www.entity.*"%>
<jsp:useBean id="stockinfo" scope="request"  class="com.jyw.www.entity.StockInfo"/>
<jsp:setProperty name="stockinfo" property="*"/>
```

```
<jsp:useBean id="stoockinfoDAO" scope="session"
                    class="com.jyw.www.dao.StockInfoDAO"/>
```

使用useBean动作产生StockInfoDAO类型的实例stockinfoDAO

再在 searchStock.jsp 中添加查询结果的显示部分。

示例 5-12　searchStock.jsp 中显示查询结果的部分代码如下。

```
<TABLE border=1>
<%
        List<StockInfo>goodsList=stockinfoDAO.findByExample(stockinfo);
        Iterator it=goodsList.iterator();
        while (it.hasNext()) {
%>
    <TR ALIGN=CENTER>
<%
          for (int j=0; j<4; j++) {
              StockInfo goods=(StockInfo) it.next();
%>
    <TD>
        <TABLE border=1>
        <TR><TD colspan="4">
          <img src="/images/<%=goods.getImage()%>" height=220  width=180>
        </TD></TR>
        <TR ALIGN=CENTER><TD colspan="4"><%=goods.getId()%></TD></TR>
        <TR ALIGN=CENTER><TD colspan="4"><%=goods.getName()%></TD></TR>
        <TR><TD><%=goods.getSize()%></TD>
            <TD><%=goods.getColor()%></TD>
            <TD>￥<%=goods.getPrice()%></TD>
            <TD><%=goods.getStock()%></TD>
        </TR>
        </TABLE>
        </TD>
<%
          if (!it.hasNext())
                    break;
        }
%>
        </TR>
<%
        }
%>
</TABLE>
```

以实体Bean实例stockinfo为入口参数，调用示例5-9中的findByExample()方法

在无任何查询条件的情况下，searchStock.jsp 的运行结果为显示所有库存，如图 5-12 所示。

以商品编号为 JYW12CKN1 和尺码为 M 作为条件的查询结果如图 5-13 所示。



图 5-12　首次进入 searchStock.jsp 页的效果示意图



图 5-13　组合条件查询结果示意图

5.5　知识扩展

5.5.1　JavaBean 的范围

在 5.4.3 小节中，使用<jsp:useBean>动作标签元素来声明一个在某一个范围内可存取的 JavaBean，这个范围可以是 application、session、request 和 page 中的一个，并通过 scope 来设置。

page 是默认的 JavaBean 范围，这种范围的 JavaBean 只能在实例化它的页面中使用，此时，JSP 动作定义的 JavaBean 实例会被保存到 PageContext 对象中。

当属性 scope 的值为 request 时，JSP 动作定义的 JavaBean 实例就会存储到 ServletRequest 对象中。在这种情况下，此实例可以被包含它的 JSP 页面所调用，不同的语句可以共享同一个 request 对象，请求结束，该实例的生命周期即结束。使用 jsp:include、jsp:forward 或者 RequestDispatcher 的 include()方法或者 forward()方法时，两个 JSP 页面或者 Servlet 页面之间将会共享这个 JavaBean 对象。

当属性 scope 的值为 session 时，JSP 动作定义的 JavaBean 实例就会存储到 HttpSession 对象中，可通过 HTTP 请求使用。该 JavaBean 实例会在整个会话(session)过程中存在。

当属性 scope 的值为 application 时，JSP 动作定义的 JavaBean 实例就会被存储到 ServletContext 对象中。这意味着 JavaBean 实例可以被运行在应用服务器的在相同生命周期中的任何对象(例如 JSP 和 Servlet)调用。

5.5.2　使用 jsp:setProperty 关联 Bean 属性和 request 参数

在 5.4.4 小节中使用<jsp:setProperty>动作在表单的参数和实体 Bean 的属性间做了批量关联。<jsp:setProperty>动作还有其他多种用法，读者可以根据业务逻辑实现的需要进行适当地选择。表 5-1 概述了使用<jsp:setProperty>元素来设置 JavaBean 组件的属性的各种方法。

表 5-1　用 String 值给 Bean 的属性值赋值

值的来源	元 素 语 法
String 常量	<jsp:setProperty name="beanName" property="propName" value="string constant"/>
Request 参数	<jsp:setProperty name="beanName" property="propName" param="paramName"/>
匹配 bean 属性的 Request 参数	<jsp:setProperty name="beanName" property="propName"/> 或者 <jsp:setProperty name="beanName" property="*"/>
表达式	<jsp:setProperty name="beanName" property="propName" value="expression"/> 或者 <jsp:setProperty name="beanName" property="propName"> <jsp:attribute name="value"> expression </jsp:attribute> </jsp:setProperty>

在使用<jsp:setProperty>元素来设置 JavaBean 组件的属性时，要注意以下几点。

(1) 在使用这个元素之前必须使用<jsp:useBean>声明此 Bean。

(2) <jsp:setProperty>中的 beanName 必须和 useBean 属性里面定义的 id 属性值相同。

(3) 在 JavaBean 组件中必须有一个相应的 setPropertyName 方法。

从一个字符串常量或者请求参数来设置属性时需要用到下表中列出的类型。如果常量或者请求(request)参数是字符串(String)类型的,Web 容器会自动把值的类型转换成属性的类型,也就是说这种转换是自动发生的,而不是由用户通过程序来控制的。这种转换如表 5-2 所示。

表 5-2　自动转换属性的类型

属性类型	在字符串值上的转换
Bean 属性	使用 setAsText(string-literal)
boolean 或 Boolean	java. lang. Boolean. valueOf(String)
byte 或 Byte	java. lang. Byte. valueOf(String)
char 或 Character	java. lang. String. charAt(0)
double 或 Double	java. lang. Double. valueOf(String)
int 或 Integer	java. lang. Integer. valueOf(String)
float 或 Float	java. lang. Float. valueOf(String)
long 或 Long	java. lang. Long. valueOf(String)
short 或 Short	java. lang. Short. valueOf(String)
Object	new String(string-literal)

1. 使用表达式设置属性

在实际的应用中设置 Bean 属性的时候,通常值都是动态的变量而非常量,这就需要使用表达式或者其他的方式来给它们赋值。

示例 5-13　使用表达式设置 StockInfo 的 size 属性

```
<jsp:useBean id="stockinfo" class="com.jyw.www.entity.StockInfo"/>
<jsp:setProperty name="stockinfo" property="size" value="
                                        <%=request.getParameter("size") %>" />
```

在示例 5-13 中使用了表达式,并且将它用在 setProperty 中,它将会接收从表单中传递过来的数据,并且用这些值来设置 JavaBean 的属性。

2. 使用 setProperty 的 param 属性

由于 request. getParameter( ) 的返回值是 String 类型,我们如果使用 request. getParameter()从表单中接收数据,再将它赋给 JavaBean 中的非 String 类型的属性,那么就需要对它进行转换。JSP 提供了一个很好的解决方法:它允许在使用请求参数设置 JavaBean 属性的时候,将属性和参数相关联,并自动执行从字符串到数字、字符和布尔类型的值的转换。从示例 5-12 中对表单的全部元素与 JavaBean 属性的关联的演示,读者应该已经体会到使用上的方便性。问题是在某些应用中,只需要关联特定的属性,这时可

以使用 param 属性来指定输入参数，被指定的请求参数的值会自动根据 JavaBean 的属性类型转换，并且将它作为 JavaBean 属性的值。

示例 5-14 使用 param 来将请求参数和 JavaBean 属性关联，并完成了示例 5-14 的功能。

示例 5-14　使用 param 关联 request 中的参数和 StockInfo 的属性

```
<jsp:setProperty name="stockinfo" property="price" param="price"/>
<jsp:setProperty name=" stockinfo " property="name" param=" name "/>
```

示例 5-14 使用 param 来分别指定了与 JavaBean 的属性相关联的请求参数，这样在执行这个 JSP 文件的时候，就会自动把各自对应的请求参数根据 JavaBean 属性类型进行转换后赋给它。如果指定的请求参数不存在，将不采取任何的行动，也就是说，系统并不传递 NULL 到相关联的属性。

如果请求参数名称和 JavaBean 的属性名称一致，还可以更进一步地将 param 去掉，例如，对于上面的例子，因为表单中的元素名称 price、size 和 JavaBean 的属性名称一致，所以可以将上面的三行修改如下。

示例 5-15　修改代码

```
<jsp:setProperty name="stockinfo" property="price" />
<jsp:setProperty name=" stockinfo " property="name" />
```

5.5.3　使用　jsp:getProperty　获取 JavaBean 的属性

要获取 JavaBean 的属性，可以使用<jsp:getProperty >动作标签。其语法规则如下：

```
<jsp:getProperty name="beanInstanceName" property="propertyName" />
```

示例 5-16　获取 StockInfo 的 color 属性

```
<jsp:getProperty name=" stockinfo " property="color"/>
```

它将会自动调用 StockInfo 的 getColor()方法，并且将获得的返回值输出到页面中。

课 后 练 习

1. 结合以前学过的静态网页设计的知识，以及第 1 章功能模块中设计的示意图，为“佳衣屋”项目设计美观而且切合主题的页面风格，并添加导航栏。

提示：参考第 4 章中 include 指令标签的用法。

2. 连锁商业企业，如深圳茂业百货、岁宝百货等，在其网站上都可以让用户获得各分店基本信息的功能。参考这些网站的用户界面设计，为“佳衣屋”项目添加美观而且实用

的门店导航功能。

要求：给出用户选择分店的接口，列出用户选择的门店的地址、联系方式等信息。

提示：参考本章任务三，完成对第 3 章中创建的 storeinfo 数据表的条件作为分店名的查询。

3. 商品的销售情况，是企业需要实时掌握的重要信息，直接影响企业的营销、生产等其他环节的决策，请你为“佳衣屋”项目添加灵活的多条件查询销售情况的功能。

要求：结合实际应用，自主分析销售情况的查询条件都有哪些？给出用户条件输入或者选择的接口，列出查询到的销售情况列表。

提示：参考本章任务四，完成对第 3 章中创建的 salesinfo 视图的多条件查询。

4. 本章学习的主要内容是对数据表的查询与显示，这是中小型 Web 项目中必不可少的功能。比如：人事管理系统中的员工信息的查询与显示，客户关系管理系统中客户信息的查询与显示，图书管理系统中书目的查询与显示等。请按照第 1 章课后练习的要求，完成项目中相应的数据表查询的功能模块。

第 6 章　商 品 入 库

本章学习要点：

- 熟练掌握 DAO 类中 save()方法的编写；
- 熟练掌握 SQL 中 INSERT INTO 语句的各种用法；
- 熟练掌握如何在项目中使用第三方组件；
- 熟练掌握使用 JSPSmartUpload 实现文件上传的方法；
- 熟练掌握 Service 类的编写方法。

6.1　任务一：新商品信息的录入

问题：每一季的新品上市前，相关信息应该被录入到数据库中以备店员进行查询或订购等其他操作，那么，使用 JSP＋JavaBean 技术，如何向数据表中添加一条记录呢？

任务目标：

采用 JSP＋JavaBean 技术，完成向商品基本信息表中添加新商品信息的操作。

技能训练：

- DAO 类的 save 方法的编写规则。
- INSERT INTO 语句的语法规则。
- JSP 中调用 DAO 类的 save 方法。

6.1.1　编写 DAO 类中的 save　方法

完成第 5 章的学习后，已经为项目的每个数据表或者视图都建立了相应的实体 Bean 和 DAO 类，并在 DAO 类中添加了多个与查询相关的方法。要完成添加新商品信息的功能，需要在与商品基本信息表对应的 DAO 类 GoodsDAO 中添加一个用来向相应数据表中添加记录的方法，一般将其命名为 save()，方法的声明示意如下：

```
void  save(Object obj)
{…}
```

save()方法中需要执行 SQL 中的 INSERT INTO 语句来完成记录的添加，语法

如下：

```
INSERT INTO 表名称 (列名 1, 列名 2,...) VALUES (值 1, 值 2,...)
```

列名的出现顺序要与值出现的顺序相对应，“值 1”应添加到以“列名 1”为名称的列中，“值 2”应添加到以“列名 2”为名称的列中，其他以此类推。

下面给出该方法的完整源代码：

示例 6-1 GoodsDAO 类中的 save()方法

```
public void save(Goods goods) {        // 实体bean Goods的实例作为入口参数
    try {
        InitialContext ctx=new InitialContext();
        DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
        Connection conn=ds.getConnection();

        PreparedStatement pstmt=conn.prepareStatement("INSERT INTO goods
        (id,name,image,price,color,size) VALUES        // 预编译 INSERT INTO语句
        (?,?,?,?,?,?)",ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_READ_ONLY);
            pstmt.setString(1, goods.getId());
            pstmt.setString(2, goods.getName());
            pstmt.setString(3, goods.getImage());
            pstmt.setFloat(4, goods.getPrice());
            pstmt.setString(5, goods.getColor());
            pstmt.setString(6, goods.getSize());        // 用户输入的商品基本信息替换SQL语句中的“?”占位符
            pstmt.executeUpdate();        // 调用executeUpdate()方法
            pstmt.close();
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
    }
}
```

注意：这里调用了 PreparedStatement 的 executeUpdate()方法，第 5 章的查询操作都是使用 executeQuery()方法。

6.1.2 编写添加商品基本信息相关的 JSP 文件

示例 6-2 addGoodsForm.jsp 中用于添加商品入库的表单部分代码如下：

```
<%@page language="java" contentType="text/html;charset=gb18030"%>
<form action="addGoods.jsp" method="post">
    <table align=center>
        <tr>
    <td colspan="2" align=center>商品基本信息</td>
    </tr>
    <tr>
```

```
        <td>编 码</td>
       <td><input type="text" name="id" id="id"></td>
     </tr>
     <tr>
         <td>名称</td>
        <td><input type="text" name="name" id="name"></td>
        </tr>
        <tr>
            <td>销售指导价</td>
            <td><input type="text" name="price" id="price"></td>
        </tr>
        <tr>
            <td>颜色</td>
            <td><input type="text" name="color" id="color">
            <input type="hidden" name="size" value="S-XL"></td>
        </tr>
        <tr>
            <td colspan=2 align=center>
            <input type="submit" value="添加"><input type="reset" value=
            "重置"></td>
        </tr>
     </table>
</form>
```

在浏览器中，addGoodsForm.jsp 程序的执行效果如图 6-1 所示。



图 6-1 商品入库基本信息填写表单示意

输入商品基本信息后，单击图 6-1 的“添加”按钮，跳转到 addGoods.jsp 页面处理，其源代码如下。

示例 6-3 addGoods.jsp

```
<%@page language="java" import="java.util.*" contentType="text/html;charset=
GB18030"%>
<%
    request.setCharacterEncoding("GB18030");
    response.setContentType("text/html;charset=GB18030");
%>
```

设置输入/输出编码方式

```
<jsp:useBean id="goods" class="com.jyw.www.entity.Goods" scope="page" />

<jsp:setProperty name="goods" property="*" />
```

将用户输入的商品基本信息与实体bean的相应属性关联

```
<jsp:useBean id="goodsDAO" class="com.jyw.www.dao.GoodsDAO" scope="page" />
<%
    goodsDAO.save(goods);
    response.sendRedirect("showGoods.jsp");
%>
```

调用GoodsDAO的save方法完成向数据表中插入一条新记录的操作

跳转到商品信息显示页

以上代码执行完毕后，可以看到 goods 数据表添加了一条新记录。

6.2 任务二：商品图片的上传

完成任务一后，商品基本信息并没有录入完整，示例 6-1 中的 INSERT INTO 语句中有 6 个占位符，但从示例 6-2 的 addGoodsForm.jsp 的表单中只接收到 5 个参数，还缺少商品的展示图片的文件名没有添加到数据库中，图片文件也没有上传至系统中，接下来将这一功能补充完整。

任务目标：

将商品的展示图片上传至项目根目录下的 images 子目录下，要求每个图片大小不能超过 200KB，并将上传图片的名称添加到 goods 数据表中。

技能训练：

- 文件上传元素的使用。
- 含有文件上传元素的表单的属性设置。
- 第三方组件的使用。
- 使用 JSPSmartUpload 完成图片的上传。
- Service 类的编写。

6.2.1 文件上载组件的使用

首先在 addGoodsForm.jsp 表单中的商品"名称"和"销售指导价"之间添加一个文件上载组件。

```
<tr>
    <td>图    片</td>
        <td><input type="file" name="file" id="file"></td>
    </tr>
```

文件上载组件的type属性为file

表单中 enctype 属性定义了表单的 MIME 编码方式，默认情况下这个编码格式是 application/x-www-form-urlencoded，不能用于文件上传；只有使用了 multipart/form-data 编码方式时，才能传递文件数据，所以含有文件上载组件的表单应设置为 enctype="multipart/form-data"，表单的 method 应设置为“POST”。将示例 6-2 中的 addGoodsForm.jsp 的表单属性设置做如下修改：

```
<%@page language="java" contentType="text/html;charset=gb18030"%>
<form action="addGoods.jsp"  method="post"  enctype="multipart/form-data">
```

表单的传输方式必须是post

带文件上传功能的表单的MIME编码方式必须是multipart/form-data

重新执行 addGoodsForm.jsp，其在浏览器中的显示效果如图 6-2 所示。

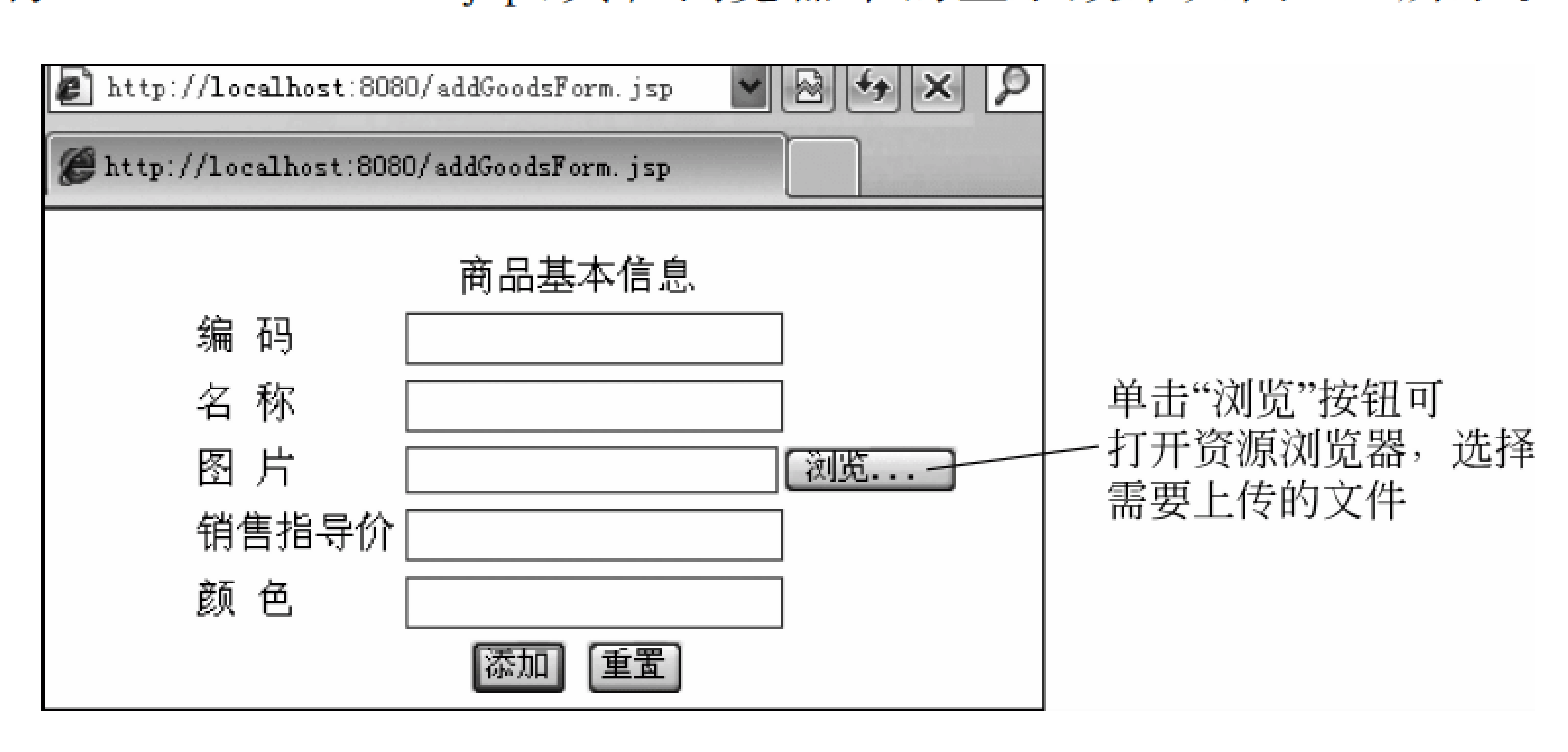


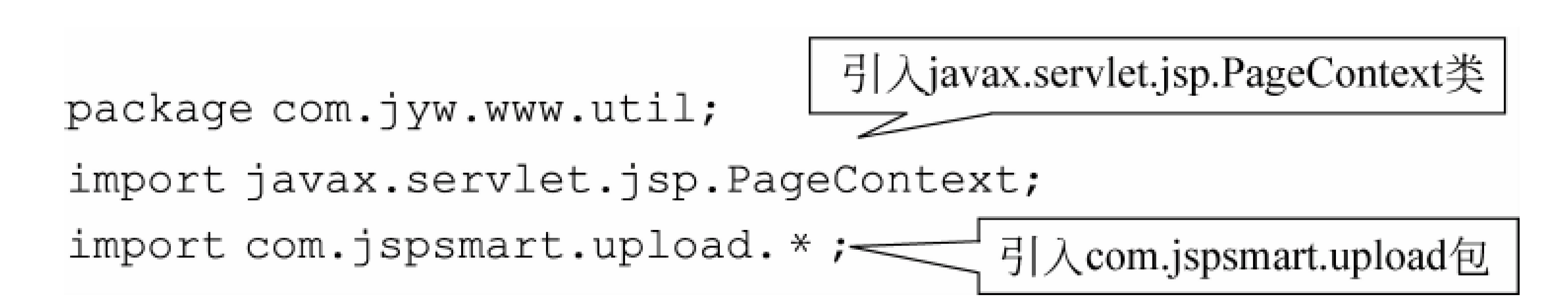
图 6-2　添加了文件上载组件后的商品入库基本信息填写页面

6.2.2　第三方组件 JSPSmartUpload 的使用

在项目的开发过程中常常需要使用第三方组件来提高开发效率。第三方组件就是由 Oracle 以外的公司开发且已经封装好某些既定功能的组件，只需要正确地进行使用，便可实现这些功能。jspSmartUpload 就是一个开源的文件上传、下载组件，通过使用 jspSmartUpload，可以很方便地实现文件的上传，并设置上传文件的存放目录和限制上传文件的大小和类型。要使用 jspSmartUpload 组件，需要下载 jspSmartUpload 的 jar 文件(请读者自行下载)。下载完成后将 jar 文件放入到项目根目录下的 Web-INF/lib 子目录下。为了使项目源代码能够编译通过，还要将其添加到项目的 Build Path 中。使用 MyEclipse 向项目的 Build Path 中添加 jar 文件的方法如图 6-3～图 6-6 所示。

下面编写 FileManager.java，其 uploadFile 方法中使用了 jspSmartUpload 的方法来实现文件上传。可将 FileManager.java 文件创建到项目的 util 包中。

示例 6-4　FileManager.java




```
package com.jyw.www.util;
import javax.servlet.jsp.PageContext;
import com.jspsmart.upload.*;
```

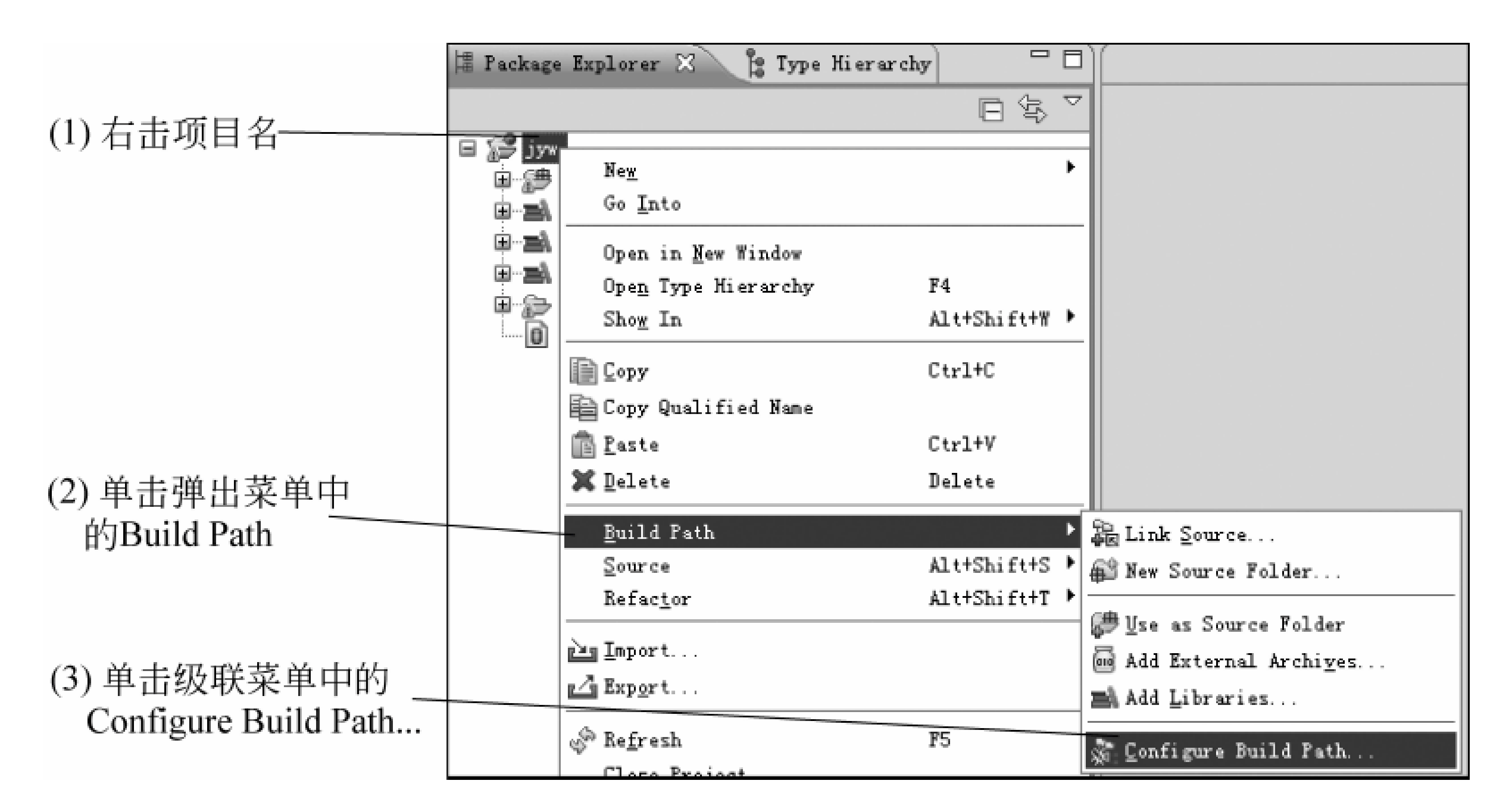


图 6-3　MyEclipse 向项目的 Build Path 中添加 jar 文件

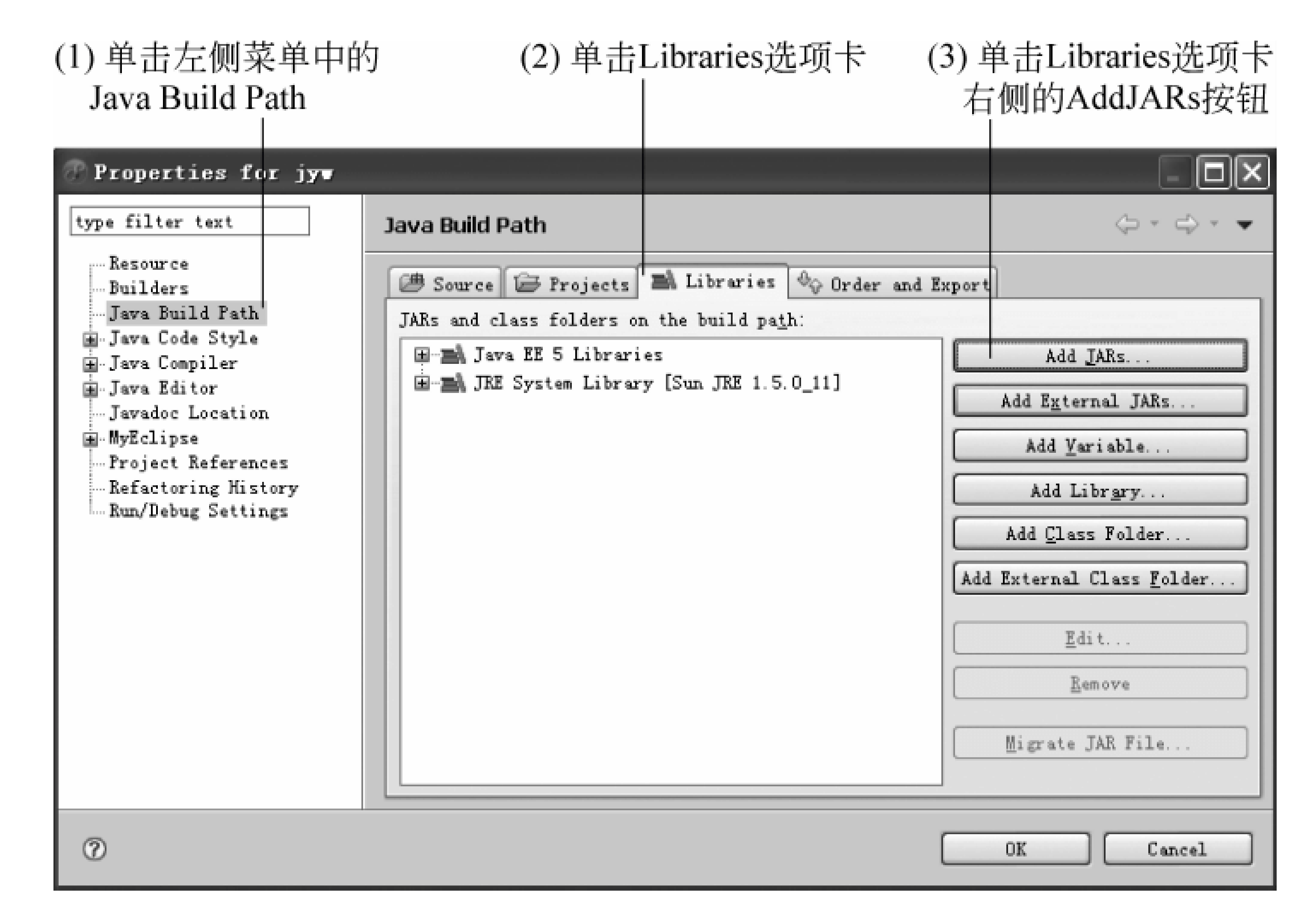


图 6-4　打开 Libraries 选项卡

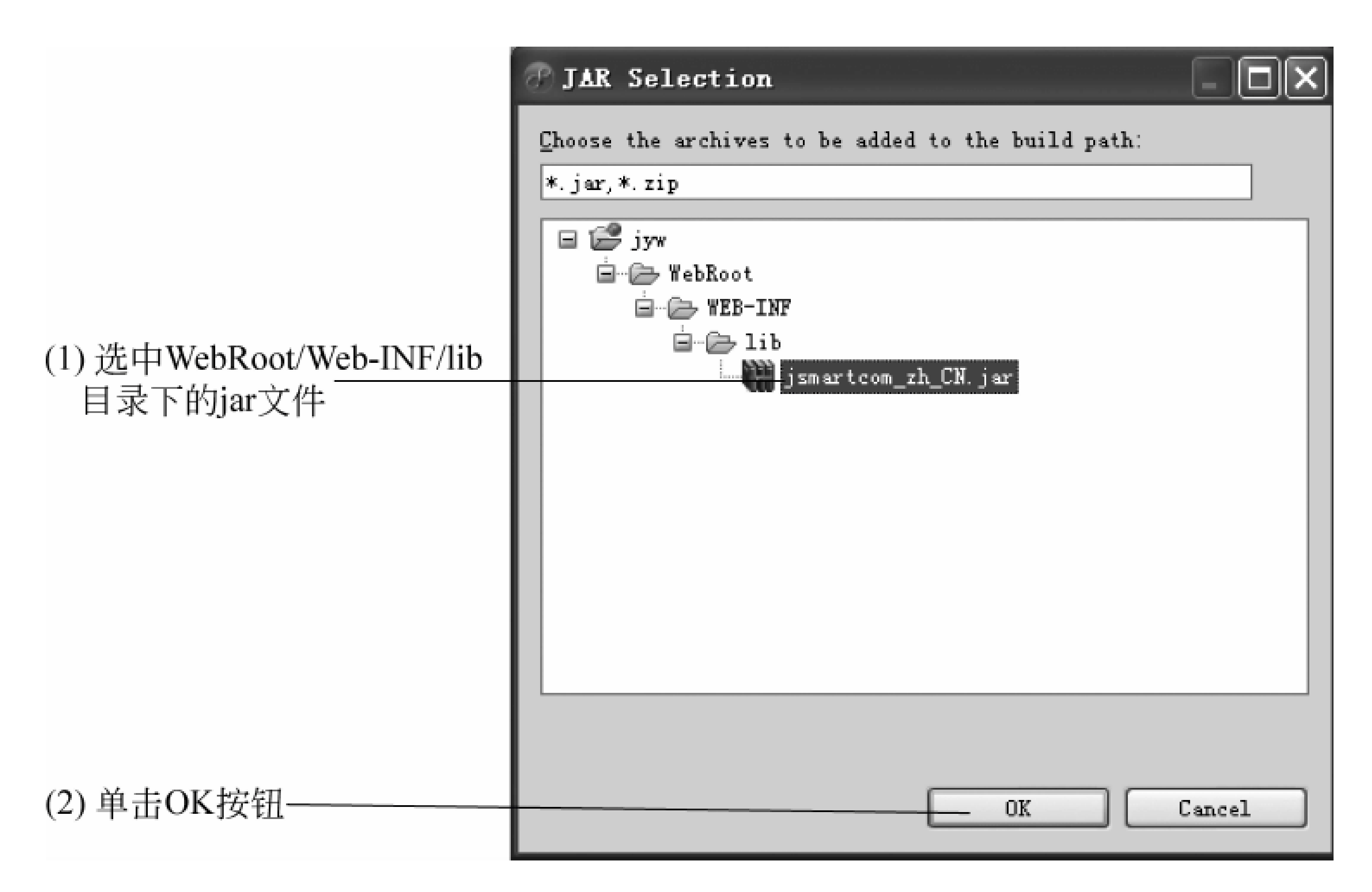


图 6-5　选择需要添加的 jar 文件

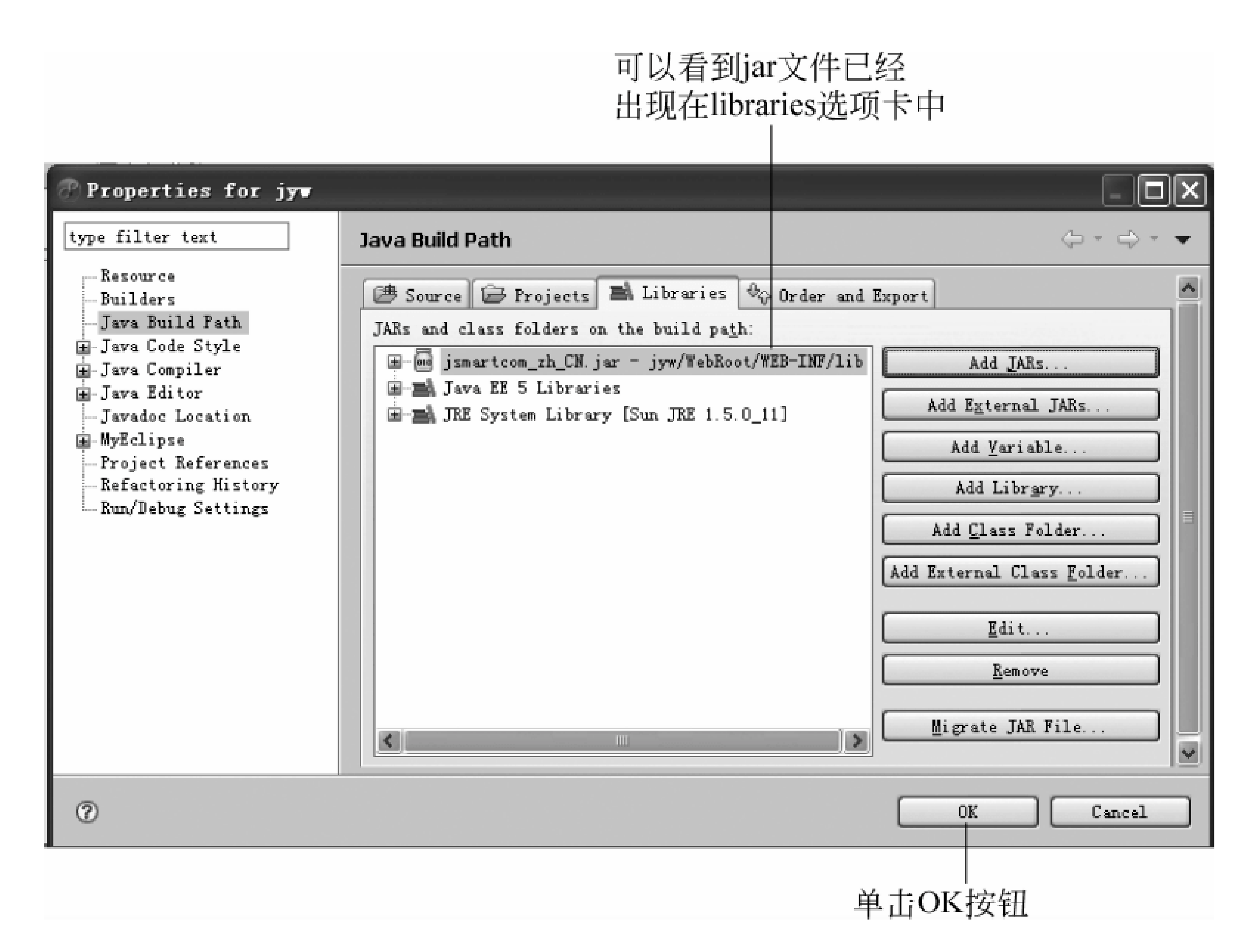


图 6-6　完成 jar 文件的添加

```
import com. jyw.www.entity. * ;

public class FileManager {
   public void uploadFile (PageContext pageContext)
   {
     SmartUpload smartupload=new SmartUpload();      创建SmartUpload实例

     try {
         smartupload.initialize(pageContext);       初始化

         smartupload.setTotalMaxFileSize(200000);   设置上传文件的总的大小上限为200KB

         smartupload.upload();                      上传文件

         smartupload.save("/images");               设置上传文件存放目录为/images
     } catch (Exception e) {
              e.printStackTrace();
     }
   }

}
```

6.2.3　编写 GoodsService 类

要完成本任务，涉及两个“动作”，一是要完成文件的上传；二是向数据表 goods 中添加一条新记录，这两个“动作”分别由示例 6-4 中 FileManager 类的 uploadFile()方法和示

例 6-1 中 GoodsDAO 中的 save()方法完成。对于这种较为复杂的业务逻辑，需要为项目引入一个业务层(service)来处理。

为项目添加 com. jyw. www. service 包，在包中新建 GoodsService 类，源代码如下。

示例 6-5 GoodsService. java

```
package com.jyw.www.service;

import javax.servlet.jsp.PageContext;
import com.jyw.www.dao.GoodsDAO;
import com.jyw.www.entity.Goods;
import com.jyw.www.util.FileManager;
import com.jspsmart.upload.*;

public class GoodsService {
    public void save(PageContext pageContext) throws Exception
    {
        SmartUpload smartupload=new FileManager().uploadFile(pageContext);   // 上传文件
        Goods goods=new Goods();
        File file=smartupload.getFiles().getFile(0);
        goods.setId(smartupload.getRequest().getParameter("id"));
        goods.setName(smartupload.getRequest().getParameter("name"));
        goods.setPrice(Float.parseFloat(smartupload.getRequest().getParameter
        ("price")));
        goods.setColor(smartupload.getRequest().getParameter("color"));
        goods.setSize(smartupload.getRequest().getParameter("size"));
        goods.setImage(file.getFileName());
        // 获取通过表单元素传递的商品信息（以上 File … setImage 各行）
        new GoodsDAO().save(goods);   // 向数据表中插入新记录
    }
}
```

表单中设置 enctype="multipart/form-data"后，除了 file 类型组件外，其他元素的 value 通过 request. getParameter 都不能取得，在示例 6-5 中，使用了 JSPSmartUpload 组件来解决这一问题，使用了 smartupload. getRequest(). getParameter("xxxx")来获取表单元素的值。

在添加了 Service 层后，addGoods. jsp 也需要做相应的修改，修改后代码如下：

示例 6-6 修改后的 addGoods. jsp

```
<%@page language="java" import="com.jyw.www.service.* "
                         contentType="text/html;charset=gb18030"%>

<%
    GoodsService gs=new GoodsService();
    gs.save(pageContext);   // 调用GoodsService类的save()方法
    response.sendRedirect("showGoods.jsp");
%>
```

至此，文件将被上传到/images 目录下，文件名存放在数据表 goods 中。

6.3　任务三：分店批量申请进货

新货上市时，分店需要从总店进货，这一过程由分店向总店发起“订单申请”开始，即向存储订单信息的 orders 数据表中添加新记录，所以也是用数据表的插入操作来完成。但是如果需要一次进货的种类很多，而每次仅能申请一种货品的进货，会使操作的效率很低，用户体验不理想。下面与读者探讨如何“批量申请进货”。

任务目标：

分店店员进入进货页，可看到所有总店库存商品的列表，勾选需要进货的条目，并在其后输入进货量，单击“提交”按钮后，完成批量进货的申请。

技能训练：

MySQL 批量添加记录语句的使用方法。

批量进货申请功能的底层实现是向 orders 表中添加多条记录，这当然可以通过多次调用 INSERT INTO 语句来完成，但这样会增加服务器的负荷，因为每执行一次 SQL 语句，服务器都要对 SQL 语句进行分析、优化等操作。

MySQL 提供了另一种解决方案，即使用一条 INSERT INTO 语句来插入多条记录。

```
INSERT INTO table (colum1, colum2,...,column)  VALUES
                  (value11,value12,...value1n),
                  (value21,value22,...value2n),
                  ⋮
                  (valuen1,valuen2,...valuenn)
```

注意：上述批量插入的 INSERT INTO 语句不是标准的 SQL 语句，仅对 MySQL 数据库的访问有效。

6.3.1　重载 OrdersDAO 类中的 save　方法

在完成本章任务一的过程中，为 GoodsDAO 类添加了一个用来插入一条记录的 save()方法，可以看做是每个 DAO 类的“标准配置”。与存放订单信息的 orders 数据表相对应的 DAO 类 OrdersDAO 类也应有一个 save(Orders orders)方法，用来向 orders 表中插入一条“订单”记录。读者可参照任务一的步骤自行完成。

但任务二的要求是一次插入多条记录。下面在 OrdersDAO 类中重载 save()方法。由于入口参数为批量订单信息，所以使用了 ArrayList 类型来作为入口参数类型，源代码如下。

示例 6-7　OrdersDAO 中的 save(ArrayList<Orders> orderslist)方法

```
public void save(ArrayList<Orders>orderslist) {
```

```
        InitialContext ctx;
        try {
          ctx=new InitialContext();
          DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
          Connection conn=ds.getConnection();
          Statement stmt=conn.createStatement();
          java.util.Date ud=new java.util.Date();
          java.sql.Timestamp stp=new java.sql.Timestamp(ud.getTime());
          String sql="INSERT INTO orders
                    (goodsid,size,quantity,storenum,applydate,orderstatus) VALUES";
```

一次插入多条记录的INSERT INTO语句的前半部分

因为订货条目数不确定，所以使用循环来遍历orderslist，并将订货信息逐条加入INSERT INTO语句的VALUES部分

```
          for (Orders order : orderslist)
          {
            sql=sql+" ('"+order.getGoodsid()+"','"+order.getSize()
                +"',"+order.getQuantity()+","+order.getStorenum()
                +",'"+stp+"',"+"'待处理'"+")"+","
          }

          sql=sql.substring(0, sql.length()-1);

          stmt.executeUpdate(sql);
        } catch (Exception e) {
          e.printStackTrace();
        }
    }
```

去掉INSERT INTO语句末尾的“,”，使得语法正确

6.3.2 编写与添加批量订单相关的 JSP 文件

下面编写分店进行批量进货的 JSP 页面，该页面列出了总店所有库存商品的信息，可供分店进行进货选择，并可输入订货量。

示例 6-8 orderForm.jsp

```
<%@page language="java" import="java.util.*,com.jyw.www.dao.*,
    com.jyw.www.entity.*"contentType="text/html;charset=gb18030"%>
<HTML>
    <BODY>
    <CENTER>
    总店库存商品列表
    <FORM ACTION="ordersSubmit.jsp" method="post" name="form2">
    <TABLE border=1>
        <TR>
          <TD align=center><B>选择</B></TD>
          <TD align=center><B>图片</B></TD>
          <TD align=center><B>商品编号</B></TD>
```

```
            <TD align=center><B>尺码</B></TD>
            <TD align=center><B>单价</B></TD>
            <TD align=center><B>库存量</B></TD>
            <TD align=center><B>进货量</B></TD>
        </TR>
    <%
      StockInfoDAO stockinfodao=new StockInfoDAO();
      List<StockInfo>stocklist=stockinfodao.findByStoreName("佳衣屋总店华强北店");
      for (StockInfo stockinfo : stocklist) {
    %>
        <TR ALIGN=CENTER>
            <TD><input type="checkbox"name="goodsid" value="<%=stockinfo.getId()%>">
            </TD>
            <TD><img width=60 height=60 src="/images/<%=stockinfo.getImage()%>">
            </TD>
            <TD><%=stockinfo.getId()%></TD>
            <TD><%=stockinfo.getSize()%></TD>
            <TD><%=stockinfo.getPrice()%></TD>
            <TD><%=stockinfo.getStock()%></TD>
            <TD><input type="text"name="<%=stockinfo.getId()%>quan"size=6></TD>
        </TR>
        <input type="hidden" value="<%=stockinfo.getSize() %>"
                              name="<%=stockinfo.getId() %>size">
      <%
        }
      %>
      </TABLE>
        <input type="hidden" value="1" name="storenum">
        <input type="submit" value="确定进货">
      </FORM>
      </CENTER>
      </BODY>
</HTML>
```

从StockInfo视图中查找“佳衣屋总店华强北店”的库存信息

采用复选框，使得分店店员可一次选择多条商品

“进货数量”文本框的名字中包含了该条“商品编码”，以便于处理商品与进货数量的对应关系

不需要用户干涉的参数，用“隐藏”类型的元素来传递

分店默认为“一分店”。在第9章中我们将学习如何自动识别当前用户所属分店，再将此功能进一步完善

该页的执行效果如图 6-7 所示。

勾选需要进货的商品，并填写进货量后，单击“确定进货”按钮，表单数据将提交给 ordersSubmit.jsp 文件来处理，该文件的源代码如示例 6-9 所示。

示例 6-9　ordersSubmit.jsp

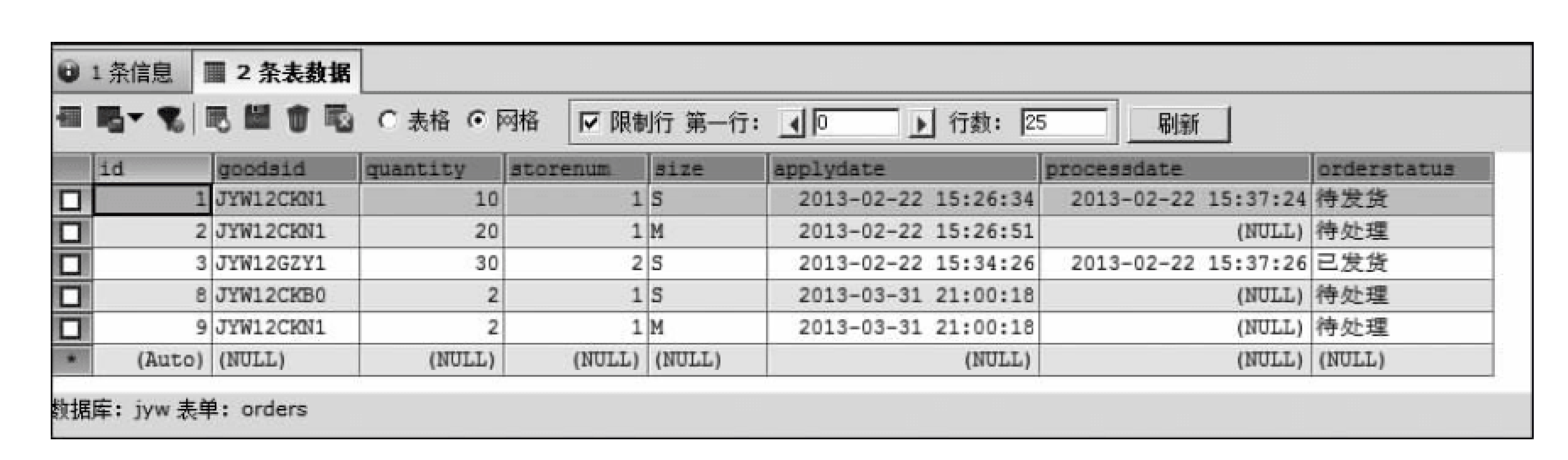
```
<%@page language="java"
    import="java.util.*,com.jyw.www.dao.*,com.jyw.www.entity.*"
    contentType="text/html;charset=gb18030"%>
<%
```



图 6-7　用户批量订货页面的显示效果

```
    OrdersDAO ordersDao=new OrdersDAO();
    String[] goodsid=request.getParameterValues("goodsid");
    if (goodsid !=null) {
        ArrayList<Orders>orderlist=new ArrayList<Orders>();
        int storenum=Integer.parseInt(request.getParameter("storenum"));
        for (int i=0; i<goodsid.length; i++) {
            Orders order=new Orders();
            order.setGoodsid(goodsid[i]);
            order.setSize(request.getParameter(goodsid[i]+"size"));
            order.setQuantity(Integer.parseInt(request.getParameter
            (goodsid[i]+"quan")));
            order.setStorenum(storenum);
            orderlist.add(order);
        }
        ordersDao.save(orderlist);
        response.sendRedirect("orderSubmitSuccess.jsp");
    } else {
        response.sendRedirect("orderSubmitFail.jsp");
    }
%>
```

单击“提交”按钮后，将完成订单的提交，图 6-8 为完成订货记录添加后 orders 数据表中的记录，其中 id 为 8 和 9 的记录为一次添加的两条新订货信息。



id	goodsid	quantity	storenum	size	applydate	processdate	orderstatus
1	JYW12CKN1	10	1	S	2013-02-22 15:26:34	2013-02-22 15:37:24	待发货
2	JYW12CKN1	20	1	M	2013-02-22 15:26:51	(NULL)	待处理
3	JYW12GZY1	30	2	S	2013-02-22 15:34:26	2013-02-22 15:37:26	已发货
8	JYW12CKB0	2	1	S	2013-03-31 21:00:18	(NULL)	待处理
9	JYW12CKN1	2	1	M	2013-03-31 21:00:18	(NULL)	待处理
(Auto)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

数据库: jyw 表单: orders

图 6-8　orders 数据表中出现新记录

6.4　知 识 扩 展

6.4.1　使用 INSERT INTO 语句插入记录的其他用法

在完成本章任务的过程中,使用了 SQL 的 INSERT INTO 语句向数据表格中插入新记录。INSERT INTO 语句还可以根据不同的应用情况,采用不用的语法格式。

本章第一节中给出 INSERT INTO 语句,采用了列名称来指出需要写入的列,这样的设计使程序有较好的可读性。为了提高执行效率,列名也可以由列序号来代替,语法如下:

```
INSERT  INTO 表名称 (列 1 序号, 列 2 序号,…) VALUES (值 1, 值 2,…)
```

采用上述语法,示例 6-1 中的 INSERT INTO 预编译语句可做如下改写:

```
PreparedStatement pstmt=conn.prepareStatement("INSERT INTO goods (1,2,3,4,5,6)
                        VALUES(?,?,?,?,?,?)",ResultSet.TYPE_SCROLL_
                        INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
```

如果新插入的记录的每一列都要写入,即值的个数与表的列数相同,还可以省略列名/列序号部分,语法如下:

```
INSERT  INTO 表名称 VALUES (值 1, 值 2,…)
```

采用上述语法,示例 6-1 中的 INSERT INTO 预编译语句可做如下改写:

```
PreparedStatement pstmt=conn.prepareStatement("INSERT INTO goods VALUES
                        (?,?,?,?,?,?)",ResultSet.TYPE_SCROLL_INSENSITIVE,
                        ResultSet.CONCUR_READ_ONLY);
```

6.4.2　使用 INSERT INTO 语句进行表复制

开发 Web 应用程序的过程中,会遇到需要将一个数据表的全部或者部分数据复制到另外一个数据表中。

如果两个数据表的列一致,并且需要复制全部数据,可以采用下述语句:

```
INSERT  INTO 目标表 SELECT * FROM  来源表
```

如果两个数据表的列一致,但仅需要复制满足某种条件的全部数据,可以采用下述语法:

```
INSERT  INTO 目标表 SELECT * FROM  来源表 WHERE 条件
```

如果只需要复制指定的列,可以采用下述语句:

```
INSERT INTO 目标表(列 1,列 2,...)SELECT 列 1,列 2,...FROM 来源表 WHERE 条件
```

注意:列的顺序必须一致。

为了避免复制后出现重复的记录,可以采用下述语句:

```
INSERT INTO 目标表(列 1,列 2,...)SELECT 列 1,列 2,...FROM 来源表
    WHERE NOT  EXSIT (select * from 目标表 WHERE 条件);
```

读者应根据应用的实际情况优化 SQL 语句的使用。

6.4.3 executeQuery 和 executeUpdate 方法的比较

在第 5 章中,执行 SQL SELECT 语句使用了 Statement/PrepareStatement 接口的 executeQuery()方法,这个方法用于执行 SELECT 语句,产生单个结果集。

本章中执行 SQL INSERT INTO 语句使用了 executeUpdate()方法,这一方法用于执行 INSERT INTO 以及后续章节中的 UPDATE 或 DELETE 语句以及 SQL DDL(数据定义语言)语句,如 CREATE TABLE 和 DROP TABLE。executeUpdate()方法的返回值类型为整型,值为语句执行后受到影响的行数。

读者可以这样理解和记忆:语句执行后,数据表内容不变,则使用 executeQuery(),语句执行后,数据表的内容有更新,则使用 executeUpdate()。

课 后 练 习

1. 为"佳衣屋"项目实现"添加新用户"的功能。

要求:给出为系统添加新用户的 UI 页面,在该页中可以输入用户的用户名、密码、真实姓名,选择该用户所属分店,选择该用户的权限(管理员/总店店员/分店店员)。单击"提交"按钮后,将用户名、密码、真实姓名、选择所属分店和使用权限添加到第 3 章创建的 employee 数据表中。

提示:参考本章任务一,完成向第 3 章创建的 employee 数据表中插入一条新记录。

2. 为 stock 库存信息表的数据库操作的 StockDAO 类添加 save 方法。

要求:为 StockDAO 类添加 save()方法,该方法完成向第 3 章中设计的 stock 数据表

中添加一条记录的功能。

提示：参考本章示例 6-1，为 StockDAO 类添加 save()方法。

3. 本章主要学习了如何向数据表中的插入新记录，这也是中小型 Web 项目中必不可少的功能。比如：人事管理系统中的员工信息（批量）的录入，客户关系管理系统中客户信息（批量）的录入，图书管理系统中书目信息的（批量）的录入等。请结合第 1 章课后练习的要求，完成项目中相应的向数据表插入新记录的功能模块。

第 7 章　商品信息的修改和删除

本章学习要点：

- 熟练掌握 DAO 类中 update()方法的编写方法；
- 熟练掌握通过 URL 传递参数；
- 熟练掌握通过表单隐藏元素传递参数；
- 熟练掌握 DAO 类中 delete()方法的编写方法。

问题：第 5 章中为系统添加了新商品的功能。商品信息在人工录入时很可能出现错误，出现错误后系统要提供修改的功能，以便更正。这个功能是如何实现的呢？

7.1　任务一：商品基本信息的修改

任务目标：

添加“修改”按钮，当单击“修改”按钮或超链接时进入修改页面，该条商品的原有信息出现在相应元素中，用户重新输入需要修改的信息后单击“提交”按钮，提交修改的信息。

技能训练：

- DAO 类中 update()方法的编写规则。
- JSP 中通过 URL 或隐藏元素来传递参数。

7.1.1　编写 DAO 类中的 update　方法

为了完成 goods 数据表中记录信息的修改，首先要为 GoodsDAO 类添加 update()方法，在此方法中需要用到 SQL UPDATE 语句来指明需要修改满足什么条件的记录，修改的是哪些列，改为何值，其语法规则是：

```
UPDATE 表名称 SET 列名称=新值 WHERE 列名称=某值
```

示例 7-1　GoodsDAO 类中的 update(Goods goods)方法

```
public void update(Goods goods) throws Exception {
```

```
    InitialContext ctx;
    try {
        ctx=new InitialContext();
        DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
        Connection conn=ds.getConnection();
        PreparedStatement pstmt=conn.prepareStatement (" UPDATE goods SET color=?,
                            name=?, size=?, price=?,image=?WHERE id=?  ");
        pstmt.setString(1, goods.getColor());
        pstmt.setString(2, goods.getName());
        pstmt.setString(3, goods.getSize());
        pstmt.setFloat(4, goods.getPrice());
        pstmt.setString(5, goods.getImage());
        pstmt.setString(6, goods.getId());
        pstmt.executeUpdate();
    } catch (Exception e) {
      e.printStackTrace();
    }
}
```

预编译UPDATE语句

执行UPDATE语句

7.1.2　在显示商品信息页添加进入修改信息页的用户入口

在第 5 章完成任务一的过程中，编写了 showGoods.jsp，用于显示商品的基本信息。要完成本章的任务一，需要在 showGoods.jsp 中为用户添加一个“功能性入口”，以便将用户导向商品信息修改页，同时传递 UPDATE 语句所需的条件部分，指明哪一条商品的信息需要修改。

通常有两种形式实现这一功能，在浏览器中分别表现为超链接和按钮。

形式一　超链接

采用超链接形式的功能入口，要配合以 URL 传递参数，格式如下：

```
xxx.jsp ?param1=value1 & param2=value2 &...& paramN=valueN
```

示例 7-2　添加了进入修改商品信息页超链接后的 showGoods.jsp

```
<%@page contentType="text/html; charset=gb18030" language="java"%>
<%@page import="com.jyw.www.entity.*,com.jyw.www.dao.*,java.util.*"%>
<HTML>
<BODY>
<CENTER>
    <TABLE border=1>
<%
    GoodsDAO goodsDao=new GoodsDAO();
    ArrayList<Goods>goodsList=goodsDao.findAll();
    Iterator<Goods>it=goodsList.iterator();
    while (it.hasNext()) {
%>
    <TR ALIGN=CENTER>
<%
```

```
    for (int j=0; j<4; j++) {
    Goods goods=(Goods) it.next();
%>
    <TD>
    <TABLE border=1>
      <TR>
        <TD colspan="4">
        <img src="/images/<%=goods.getImage()%>" height=220  width=180>
        </TD>
      </TR>
      <TR><TD colspan="3"><%=goods.getId()%></TD></TR>
      <TR><TD colspan="3"><%=goods.getName()%></TD></TR>
      <TR><TD><%=goods.getColor()%></TD>
        <TD><%=goods.getSize()%></TD>
        <TD><%=goods.getPrice()%>￥</TD>
      </TR>
      <TR><TD colspan="3">
              <a href=updateGoodsForm.jsp?id=<%=goods.getId()%>>修改</a>
          </TD>
        </TR>
      </TABLE>
    </TD>
<%
      if (!it.hasNext())
          break;
      }
%>
</TR>
<%
}
%>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

添加超链接形式的"用户功能性入口"，使用URL传递参数id

执行效果如图 7-1 所示。

形式二　按钮

采用按钮形式的功能入口，同时传递参数，又有两种方式：

- 使用表单 hidden 元素
- 使用 button 元素

(1) 使用表单 hidden 元素传递参数

```
<a href=updateGoodsForm.jsp?id=<%=goods.getId()%>>修改</a>
```

将上面的语句更改为下面的代码即可。

```
<form action="updateGoodsForm.jsp">
      <input type="hidden" name="id" value="<%=goods.getId() %>">
```

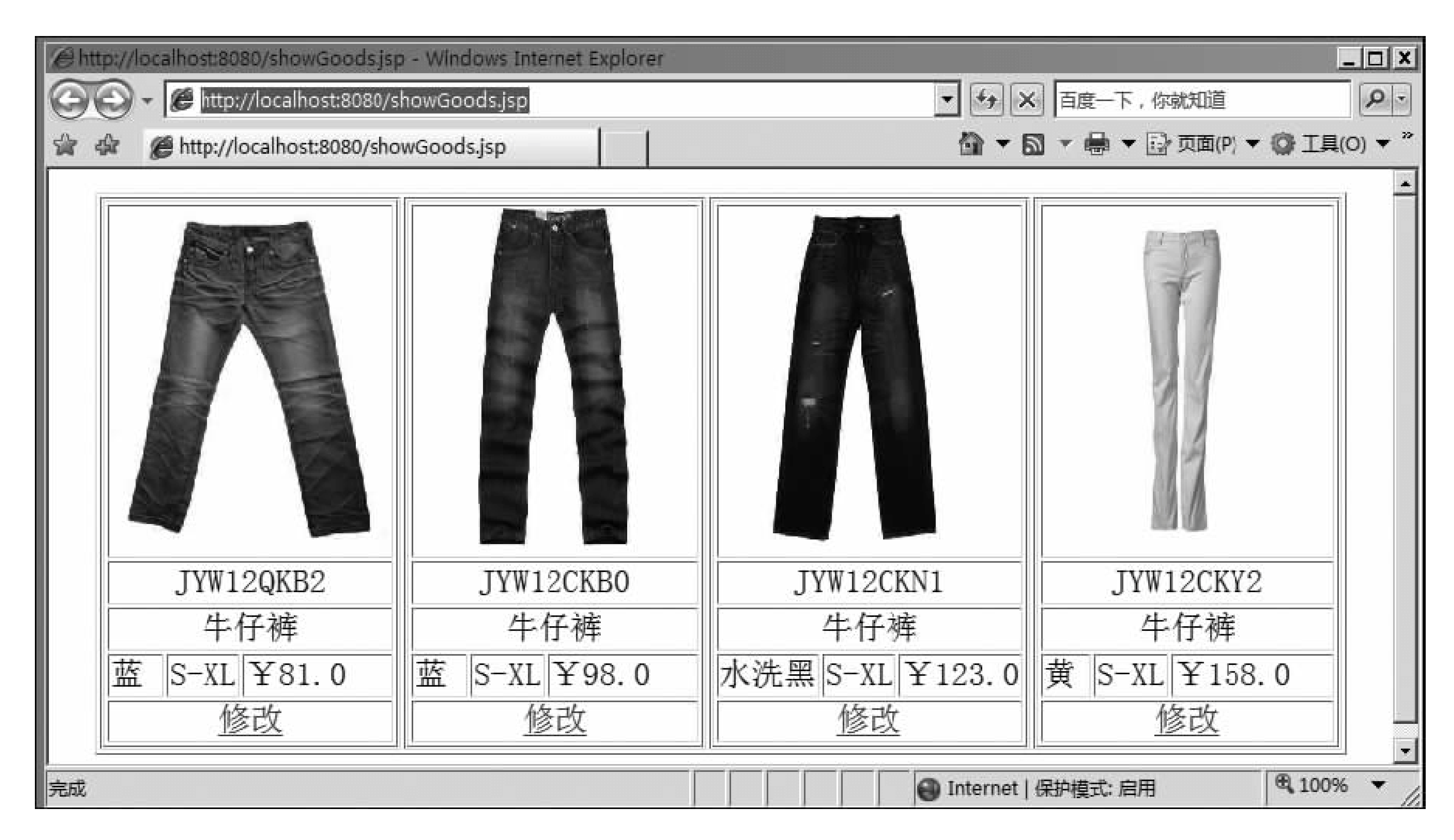


图 7-1　带超链接“修改”功能入口的商品信息显示页

```
    <input type="submit" value="修改">
</form>
```

(2) 使用 button 元素传递参数

```
<a href=updateGoodsForm.jsp?id=<%=goods.getId()%>>修改</a>
```

将源文件中如上的语句更改为下面的代码即可。

```
input type="button" name="button" value="修改" onClick="document.location.
          href=updateGoodsForm.jsp?id=<%=goods.getId()%>" /
```

采用按钮形式的功能入口的页面执行效果如图 7-2 所示。

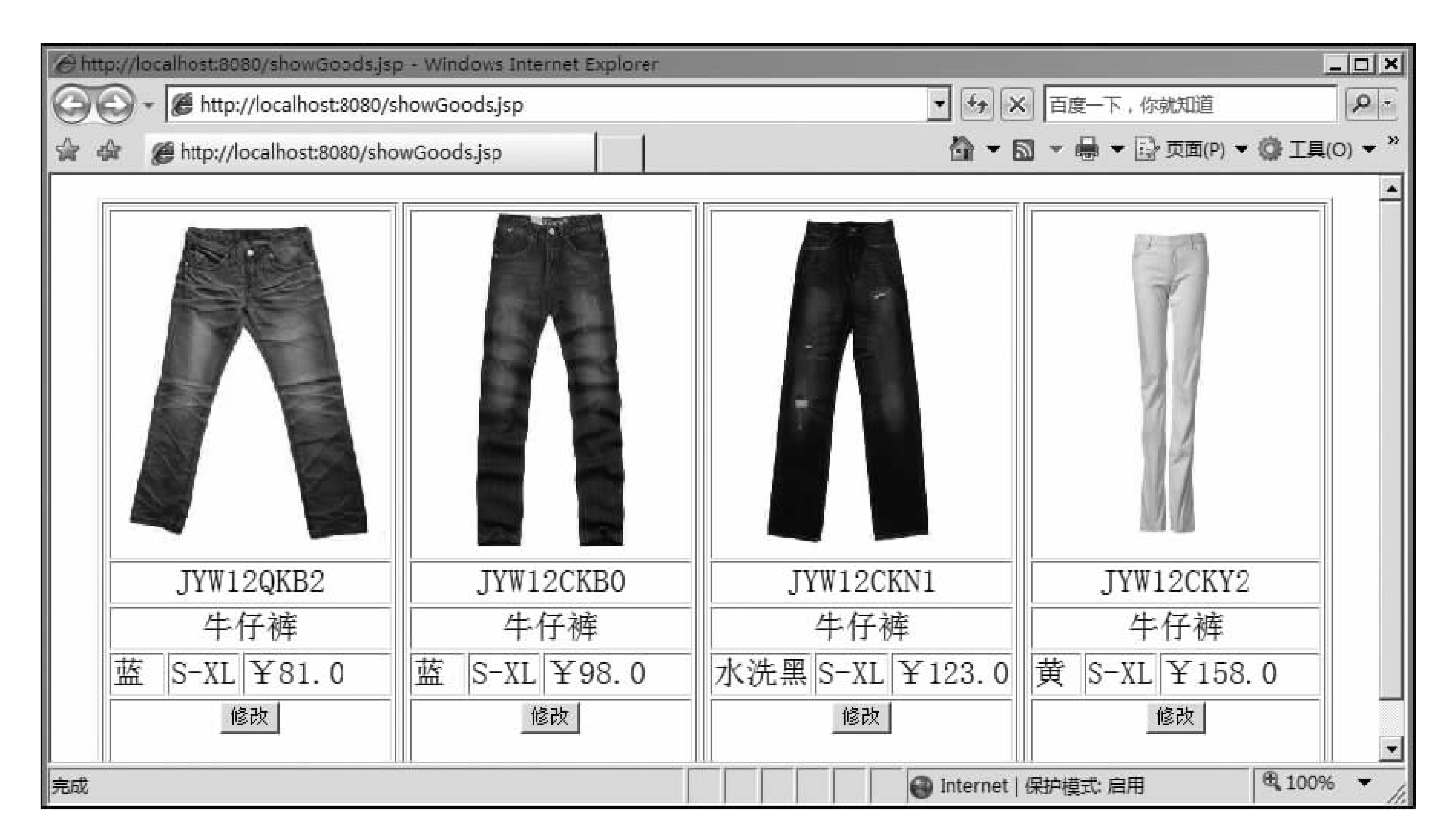


图 7-2　带按钮“修改”功能入口的商品信息显示页

7.1.3 编写修改商品信息相关的 JSP 页面

7.1.2 小节中无论是单击超链接还是按钮都将转向 updateGoodsForm.jsp，并传递了指定商品的货品编号作为参数。下面编写这一 JSP 文件。

示例 7-3 修改商品信息的 JSP 页面 updateGoodsForm.jsp

```
<%@page language="java"
    import="java.util.*,com.jyw.www.dao.*,com.jyw.www.entity.*"
    contentType="text/html;charset=gb18030"%>
<%
    GoodsDAO goodsDao=new GoodsDAO();
    Goods goods=goodsDao.findById(request.getParameter("id"));
%>
<TITLE>修改商品信息</TITLE>
<FORM action="dealUpdateGoods.jsp" method="post">
    <TABLE BORDER=1 ALIGN="center">
        <TR>
                商品信息
            </TH>
        </TR>
        <TR ALIGN=CENTER>
            <TD>
                <img src="/images/<%=goods.getImage()%>" height=220 width=180>
            </TD>
        </TR>
        <TR>
            <TD>
                编码：<%=goods.getId()%></TD>
        </TR>
        <TR>
            <TD>
                名称：
                <input type="text" name="name" id="name"
                    value="<%=goods.getName()%>">
            </TD>
        </TR>
        <TR>
            <TD>
                颜色：
                <input type="text" name="color" id="color"
                    value="<%=goods.getColor()%>">
            </TD>
        </TR>
        <TR>
            <TD>
                尺码：
```

以传递过来的参数id作为条件，获取指定编号的商品信息

将待修改的商品信息显示在相应的元素中

```
                <input type="text" name="color" id="color"
                    value="<%=goods.getSize()%>">
            </TD>
        </TR>
        <TR ALIGN=CENTER>
            <TD>
                价格:
                <input type="text" name="price" id="price"
                    value="<%=goods.getPrice()%>">
                ￥
            </TD>
        </TR>
        <TR>
                <input type="hidden" name="id" value=<%=goods.getId()%>>
                <input type="submit" value="修改">
                <input type="reset" value="重置">
            </TD>
        </TR>
    </TABLE>

</FORM>
```

执行效果如图 7-3 所示。



图 7-3　修改商品信息表单页面

输入修改后的商品信息后单击“修改”按钮，将把表单中的参数交给 dealUpdateGoods.jsp 文件进行处理，源代码如下。

示例 7-4　处理商品信息修改的 JSP 页面 dealUpdateGoods.jsp

```
<%@page language="java"
    import="java.util.*,com.jyw.www.service.*,com.jyw.www.dao.*"
    contentType="text/html; charset=gb18030"%>
<jsp:useBean id="goods" class="com.jyw.www.entity.Goods" scope="page" />
<%
    request.setCharacterEncoding("GB18030");
    response.setContentType("text/html;charset=GB18030");
%>
<jsp:setProperty name="goods" property="*" />      关联表单参数和实体bean属性
<jsp:useBean id="goodsDAO" class="com.jyw.www.dao.GoodsDAO" scope="page" />
<%
    goodsDAO.update(goods);      调用示例7-1编写的update()方法，完成商品信息的修改
    response.sendRedirect("showGoods.jsp");
%>
```

dealUpdateGoods.jsp 文件执行完毕后，将跳回 showGoods.jsp 页，读者可以看到指定商品的信息已经被修改了。

7.2　任务二：商品图片的修改

问题：完成本章任务一后，可以修改商品的一般信息了，但是如果我们需要修改其图片，又该如何处理呢？

任务目标：

在本章任务一的基础上，添加更新商品图片的功能。

技能训练：

- 使用 java.util.File 类方法删除一个文件。
- 使用 button 元素传参数。
- Java 条件运算符的使用。

7.2.1　添加修改商品图片的入口

首先在 updateGoodsForm.jsp 页面中添加修改商品图片的功能入口。

示例 7-5　添加更新图片入口后的 dealUpdateGoods.jsp

```
<%@page language="java"
    import="java.util.*,com.jyw.www.dao.*,com.jyw.www.entity.*"
    contentType="text/html;charset=gb18030"%>
<%
    GoodsDAO goodsDao=new GoodsDAO();
```

```
    String id=request.getParameter("id");
    Goods goods=goodsDao.findById(id);
    String filename=goods.getImage();
    String filechanged=request.getParameter("filechanged");
%>
```

增加一个新图片文件名的变量

```
<TITLE>修改商品信息</TITLE>
<FORM name="form1" action="dealUpdateGoods.jsp" method="post">
<TABLE BORDER=1 ALIGN="center">
    <TR><TH>商品信息</TH></TR>
    <TR ALIGN=CENTER>
    <TD>
        <img src="/images/<%=request.getParameter("filechanged") !=null ?
                    request.getParameter("filechanged") : filename%>"
                        height=220 width=180><br>



        <input name="cbutton" type="button" value="更新图片"
                    onClick="document.location.href='updateFile.jsp?id=
                    <%=id%>'"><br>


        <font color="red"><%=filechanged !=null ?filechanged : ""%><font>


        <input name="image" type="hidden"
                  value="<%=filechanged !=null ?filechanged : filename%>">
    </TD>
    </TR>
        <TR>
            <TD>
                编码：<%=id%></TD>
        </TR>
        <TR><TD>
                名称：
                <input type="text" name="name" id="name"
                    value="<%=goods.getName()%>">
            </TD></TR>
        <TR><TD>
                颜色：
                <input type="text" name="color" id="color"
                    value="<%=goods.getColor()%>">
            </TD></TR>
        <TR><TD>
                尺码：
                <input type="text" name="color" id="color"
                    value="<%=goods.getSize()%>">
        </TD></TR>
        <TR><TD>
                价格：
```

```
                <input type="text" name="price" id="price"
                    value="<%=goods.getPrice()%>">
                ￥
            </TD></TR>
            <TR><TD align="center">
                <input type="hidden" name="id" id="id" value=<%=id%>>
                <input type="submit" value="修改" />
                <input type="reset" value="重置">
            </TD></TR>
        </TABLE>
</FORM>
```

初次进入 updateGoodsForm.jsp 页时的执行效果如图 7-4 所示。



图 7-4　添加了图片更新功能入口后的修改商品信息表单页面

7.2.2　编写修改商品图片相关的 JSP 文件

单击图 7-4 中的“更新图片”按钮，转向 updateFile.jsp。

示例 7-6　updateFile.jsp

```
<%@page language="java" contentType="text/html;charset=gb18030"%>
<CENTER>
<form action="dealUploadFile.jsp" method="post" enctype="multipart/form-data">
  上传图片<input type="file" name="file">
  <input type="hidden" name="id" value='<%=request.getParameter("id")%>'>
  <input type="submit" value="上传">
</form>
```

updateFile.jsp 执行效果如图 7-5 所示。

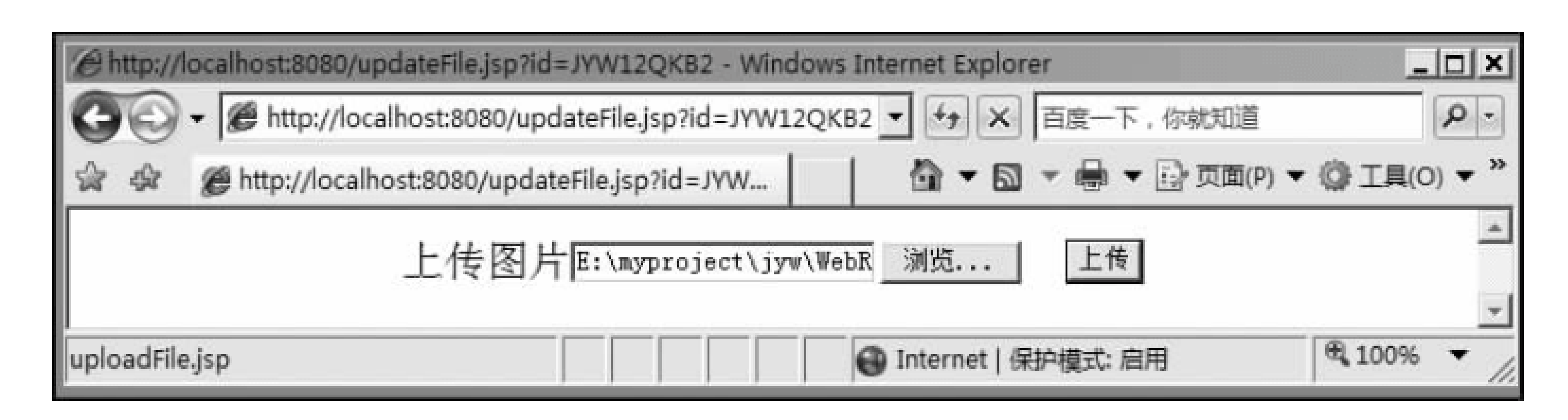


图 7-5　上传新图片的 JSP 页面

单击图 7-5 中的"浏览…"按钮可选择新图片，选中图片后，单击"上传"按钮后将转向 dealUpdateFile.jsp。

示例 7-7　dealUpdateFile.jsp

```
<%@page language="java"
    import="com.jyw.www.util.*,com.jspsmart.upload.*"
    contentType="text/html;charset=gb18030"%>
<%
    SmartUpload smartupload=new FileManager().uploadFile(pageContext);
    String filename=smartupload.getFiles().getFile(0).getFileName();
    String id=smartupload.getRequest().getParameter("id");
%>
<jsp:forward page="updateGoodsForm.jsp">
  <jsp:param name="id" value="<%=id%>" />
  <jsp:param name="filechanged" value="<%=filename%>" />
</jsp:forward>
```

dealUpdateFile.jsp 使用 JSPSmartUpload 组件完成新图片文件的上传，然后跳转回商品信息修改页。执行效果如图 7-6 所示。



图 7-6　上传新图片的 JSP 页面

单击图 7-6 中的“修改”按钮，仍然转向示例 7-4 中的 dealUpdateGoods.jsp 文件，完成商品信息的修改。

7.3 任务三：分店商品售出后存货数量的变化

问题：商品库存数量在进货和售出的时候都应该及时发生增加或者减少的变化，这个功能如何实现呢？

任务目标：

商品数量应能随着进货或者售出活动发生实时变化。

技能训练：

UPDATE 语句的灵活使用。

如果对数据表中的某列的修改是在原值的基础上进行增、减，UPDATE 语句还有一种适合这种应用的用法，其语法格式如下：

```
UPDATE 表名称 SET 列名称=新值 WHERE 列名称=表达式
```

7.3.1 编写 StockDAO 类中用于修改库存的方法

首先在与库存信息数据表 stock 相对应的 StockDAO 类中添加一个用于修改库存量的方法 updateOnOrders()，该方法在“用户订货”的情况下将减少同时满足商品编号、尺码、货品所在分店为用户提交的参数，并且当前库存量大于订货量这四个条件的 stock 数据表中记录的库存量；在“用户退货”的情况下将增加同时满足商品编号、尺码、货品所在分店为用户提交的参数的记录的库存量。源代码示例如下。

示例 7-8 StockDAO 中的 updateOnOrders()方法

```
public int updateOnOrders(Orders order) throws Exception {
    InitialContext ctx;
    try {
        ctx=new InitialContext();
        DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
        Connection conn=ds.getConnection();
        PreparedStatement pstmt=null;
        String orderstatus=new String(order.getStatus().getBytes("ISO-
        8859-1"),"gb18030");
        if (orderstatus.equals("待发货"))
            pstmt=conn.prepareStatement("UPDATE stock SET stock=stock-? WHERE
                goodsid=? and storenum=? and size=? and stock>?");
```

如果订单状态改为“待发货”，则库存数量为当前数量-订货量

```
        else if (orderstatus.equals("已退货"))
            pstmt=conn.prepareStatement("UPDATE stock SET stock=stock+?WHERE
                goodsid=?and storenum=?and size=?");
```

如果订单状态改为"已退货"，则库存数量为当前数量+订货量

```
        pstmt.setInt(1, order.getQuantity());
        pstmt.setString(2, order.getGoodsid());
        pstmt.setInt(3, order.getStorenum());
        pstmt.setString(4, order.getSize());
        pstmt.setInt(5, order.getQuantity());
        return pstmt.executeUpdate();
      } catch (Exception e) {
        e.printStackTrace();
      }
      return 0;
}
```

7.3.2　编写 OrdersDAO 类中用于修改订单状态的方法

订单状态应随着订货、退货活动进行改变，这就需要在与订单信息数据表 orders 相对应的 OrdersDAO 中添加修改订单状态的方法 updateOrderStatus()，源代码示例如下：

示例 7-9　OrdersDAO 中的 updateOrderStatus()方法

```
public void updateOrderStatus(Orders order) throws Exception {
    InitialContext ctx;
    try {
        ctx=new InitialContext();
        DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
        Connection conn=ds.getConnection();
        PreparedStatement pstmt=conn.prepareStatement("UPDATE orders SET
        orderstatus=?,processdate=?WHERE id=?");
        String orderstatus=new String(order.getStatus().getBytes("ISO-
        8859-1"),"gb18030");
        pstmt.setString(1, orderstatus);
        java.util.Date ud=new java.util.Date();
        java.sql.Timestamp stp=new java.sql.Timestamp(ud.getTime());
```

取当前系统时间

转变为SQL时间戳类型

```
        pstmt.setTimestamp(2, stp);
        pstmt.setString(3, order.getId());
        pstmt.executeUpdate();
      } catch (Exception e) {
        e.printStackTrace();
      }
}
```

将数据表orders中的订单处理时间修改为当前系统时间

7.3.3 编写 OrdersService 类中的 update 方法

订单的处理过程中会导致商品库存量的变化和订单状态的变化，涉及两个数据表，所以需要编写相应的 Service 类的 update 方法来进行处理。

示例 7-10 OrdersDAO 中的 updateOrderStatus()方法

```
public Boolean update(Orders order) throws Exception
{
    String orderstatus=new String(order.getStatus().getBytes("ISO-8859-1"),"
    gb18030");
    if(orderstatus.equals("待发货"))
    {
        if(new StockDAO().updateOnOrders(order)>0)
        {
            new OrdersDAO().updateOrderStatus(order);
            return true;
        }
        else
            return false;
    }else
    {
        new OrdersDAO().updateOrderStatus(order);
            return true;
    }
}
```

在库存商品量充足的情况下，才去改变订单的状态

7.3.4 编写订单处理相关的 JSP 文件

ordersManage.jsp 执行后将列出所有的订单信息，并给出处理订单的功能入口，源文件示例如下。

示例 7-11 ordersManage.jsp

```
<%@page language="java"
    import="java.util.*,com.jyw.www.dao.*,com.jyw.www.entity.*"
    contentType="text/html;charset=gb18030"%>
<center>
    订单列表
</center>
<table border=1 align="center" bgColor="#ffffff">
    <tr>
        <td align=center>
            <B>序号</B>
        </td>
        <td align=center>
```

```
            <B>款号 </B>
        </td>
        <td align=center>
            <B>颜色 </B>
        </td>
        <td align=center>
            <B>类别 </B>
        </td>
        <td align=center>
            <B>码数 </B>
        </td>
        <td align=center>
            <B>数量 </B>
        </td>
        <td align=center>
            <B>订购日期 </B>
        </td>
        <td>
            <B>处理日期 </B>
        </td>
        <td>
            <B>分店 </B>
        </td>
        <td align=center colspan=2>
            <B>情况 </B>
        </td>
    </tr>

    <%
        OrderinfoDAO orderinfoDao=new OrderinfoDAO();
        List<Orderinfo>orderinfoList=orderinfoDao.findAll();
        for (Orderinfo orderinfo : orderinfoList) {
            int quantity=orderinfo.getQuantity();
            String status=orderinfo.getOrderstatus();
            Date processdate=orderinfo.getProcessdate();
            int id=orderinfo.getId();
            String goodsid=orderinfo.getGoodsid();
            String size=orderinfo.getSize();
    %>

    <tr>
        <td align=center><%=id%></td>
        <td align=center><%=goodsid%></td>
        <td align=center><%=orderinfo.getColor()%></td>
        <td align=center><%=orderinfo.getName()%></td>
        <td align=center><%=size%></td>
        <td align=center><%=quantity%></td>
        <td align=center><%=orderinfo.getApplydate()%></td>
        <td align=center><%=processdate !=null ?processdate : ""%></td>
```

```
            <td align=center><%=orderinfo.getStorenum()%></td>
            <%
                if (status.equals("已发货") || status.equals("已取消")) {
            %>
            <td colspan=2 align=center>
                <%=status%>
            </td>
            <%
                } else if (status.equals("待发货")) {
            %>
            <td align=center>
                <input type="button" value="发货"
                     onClick="document.location.href='ordersConfirm.jsp? id=<%=
                    id%>&status=已发货'">
            </td>
            <%
                } else {
            %>
            <td>
                <input type="button" value="确认"
                    onClick="document.location.href='ordersConfirm.jsp?id=<%=
                    id%>&goodsid=<%=goodsid%>&size=<%=size%>&storenum=
                    0&quantity=<%=quantity%>&status=待发货'">

                <input type="button" value="取消"
                    onClick="document.location.href='ordersConfirm.jsp?id=<%=
                    id%>&status=已取消'">
            </td>
            <%
                }
            %>
            <td align=center>
                <form action="orderquery.jsp" name="form1">
                    <input type="hidden" name="stock"
                            value="<%=orderinfo.getQuantity()%>">
                    <input type="hidden" name="orderid"
                            value="<%=orderinfo.getId()%>">
                    <input type="hidden" name="goodsid"
                        value="<%=orderinfo.getGoodsid()%>">
                    <input type="hidden" name="situid" value="1">
                    <input type="hidden" name="handle" value="1">
                </form>
            </td>
            <%
                }
            %>
        </tr>
    </table>
```

执行后的效果如图 7-7 所示。



图 7-7　订单列表页

单击图 7-7 中的功能按钮，将转向 ordersConfirm. jsp 文件，其源文件示例如下。

示例 7-12　ordersConfirm. jsp

```
<%@page language="java" import="java.util. * ,com.jyw.www.service. * "
    contentType="text/html;charset=gb18030"%>
<jsp:useBean id="orders" class="com.jyw.www.entity.Orders" scope="page" />
<jsp:setProperty name="orders" property=" * " />
<%
    if (!new OrdersService().update(orders)) {
        out.print("订单确认失败,请检查库存量是否足够");
        out.print("<input type='button' value='确定'
                onclick=document.location.href='ordersManage.jsp'>");
    } else
        response.sendRedirect("ordersManage.jsp");
%>
```

单击图 7-7 中的某笔订单操作栏的“确定”按钮，将转向 ordersConfirm. jsp 文件，如果当库存量大于订货数量时，执行后自动跳转回如 ordersManage. jsp 页，相应订单的操作栏改变为“发货”按钮，如图 7-8 所示。

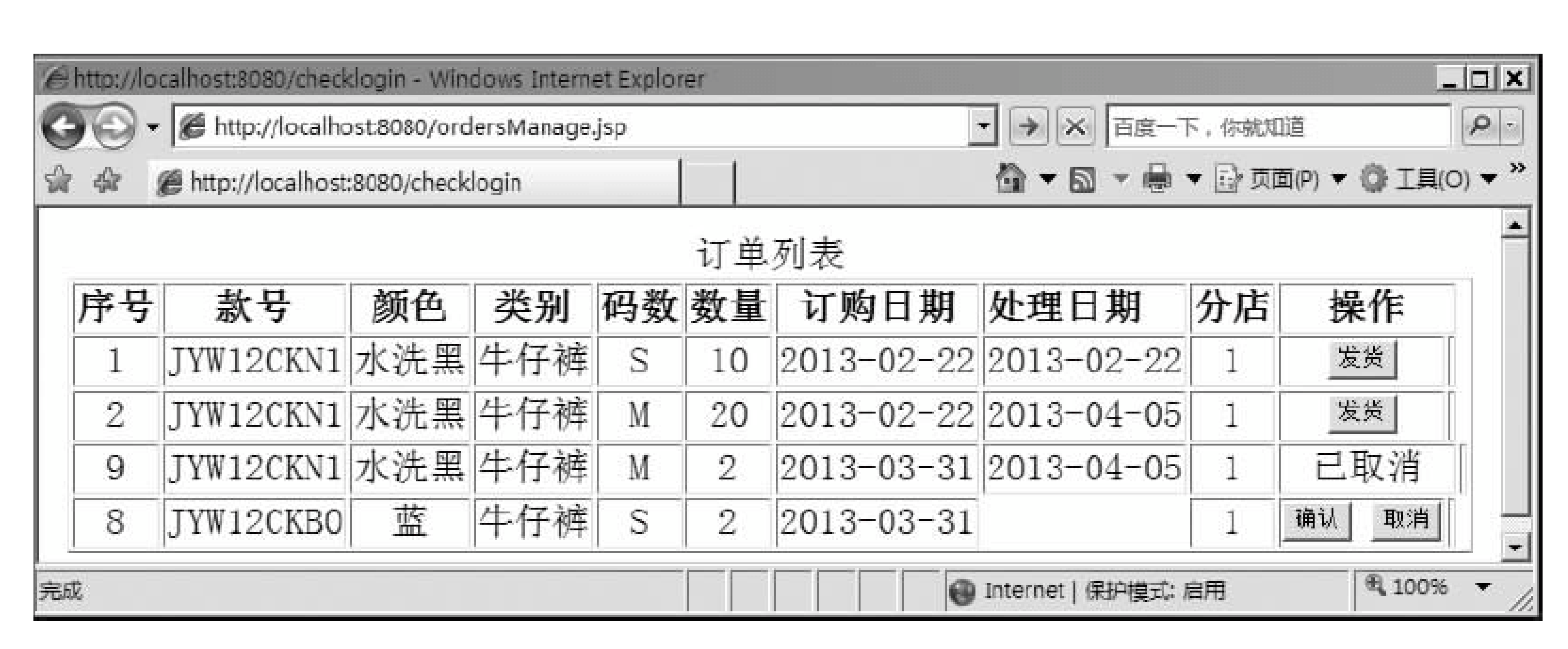


图 7-8　库存量不能满足订单要求时的提示

如果库存量小于订货数量时，OrdersService(). update()方法返回 false，这种情况下，ordersConfirm. jsp 将显示相应的提示信息，如图 7-9 所示，单击图 7-8 中的“确定”按钮将返回 ordersManage. jsp 页，操作栏保持不变。

单击图 7-7 中的某笔订单操作栏的“取消”或者“发货”按钮，将转向 ordersConfirm. jsp 文件，执行后自动跳转回如 ordersManage. jsp 页，相应订单的状态改为“已取消”或者

“已发货”，如图 7-10 所示。



图 7-9　库存量不能满足订单要求时的提示



序号	款号	颜色	类别	码数	数量	订购日期	处理日期	分店	操作
1	JYW12CKN1	水洗黑	牛仔裤	S	10	2013-02-22	2013-02-22	1	发货
2	JYW12CKN1	水洗黑	牛仔裤	M	20	2013-02-22	2013-04-05	1	已发货
9	JYW12CKN1	水洗黑	牛仔裤	M	2	2013-03-31	2013-04-05	1	已取消
8	JYW12CKB0	蓝	牛仔裤	S	2	2013-03-31		1	确认 取消

图 7-10　库存量不能满足订单要求时的提示

7.4　任务四：商品售罄后信息的删除

问题：Web 项目运行一段时期后，会积累相当规模的数据，对于不再有保存价值的数据，应当及时删除，以减少存储空间的开销。例如：随着商品的售出，商品库存数量在不断减少，当商品售罄并停产后，这条商品的信息就可以被删除了。那么项目中需要对数据表中记录进行删除的功能该如何实现呢？

任务目标：

将用户指定的商品的信息从 goods 数据表中删除。

技能训练：

DELETE 语句的使用。

7.4.1　GoodsDAO 中添加 deleteById　的方法

GoodsDAO.java 中 deleteById()方法用于删除指定编号商品的分店售罄商品的信息。

示例 7-13　GoodsDAO.java 中的 deleteById()方法

```
public void deleteById(String id) throws Exception {
```

```
    InitialContext ctx;
    try {
        ctx=new InitialContext();
        DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
        Connection conn=ds.getConnection();
        PreparedStatement pstmt=conn
                   .prepareStatement("DELETE FROM goods WHERE id=?");
        pstmt.setString(1, id);
        pstmt.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

7.4.2　编写删除商品信息相关的 JSP 页

在示例 5-5 显示商品信息页 showGoods.jsp 中添加“删除”功能入口，代码如下：

```
<input  type="button"  value="删除"
    onClick="document.location.href='delGoods.jsp?id=<%=goods.getId()%>'">
```

修改后，显示商品信息页执行效果如图 7-11 所示。



图 7-11　添加了“删除”功能入口后的商品信息显示页

单击图 7-11 中的“删除”按钮，将转向 delGoods.jsp，其源文件示例如下：

示例 7-14　delGoods.jsp

```
<%@page language="java" import="java.util.*,com.jyw.www.service.*,com.jyw.
www.dao.*"
contentType="text/html; charset=gb18030"%>
<%
new GoodsDAO().deleteById(request.getParameter("id"));
response.sendRedirect("showGoods.jsp");
%>
```

7.5 知识扩展

7.5.1 修改所有行

SQL UPDATE 语句可以不写条件部分,格式如下:

```
UPDATE table_name SET 列 1=值 1,…,列 n=值 n
```

这种情况下,数据表中所有记录的指定列都将被修改为指定的值。

7.5.2 删除所有行

SQL DELETE 语句也可以不写条件部分,格式如下:

```
DELETE FROM table_name
```

或者

```
DELETE * FROM table_name
```

这种情况下,将在不删除表的情况下删除所有的记录。表的结构、属性和索引仍保持完整。

课后练习

1. 本章任务二中完成了商品图片的更新,但是商品的原图片仍然在/Image 目录下,浪费存储空间,请在任务二基础上,补充删除原图片文件的功能。

提示:使用 java.util.File 类的相应方法。

2. 为"佳衣屋"项目添加"退货"功能。

要求:分店可向总店申请"退货",并提交退货商品的编号、尺寸、退货量等信息;总店在订单显示页中可看到分店申请的"退货"单,并可完成退货操作。

提示:参考本章任务三完成。

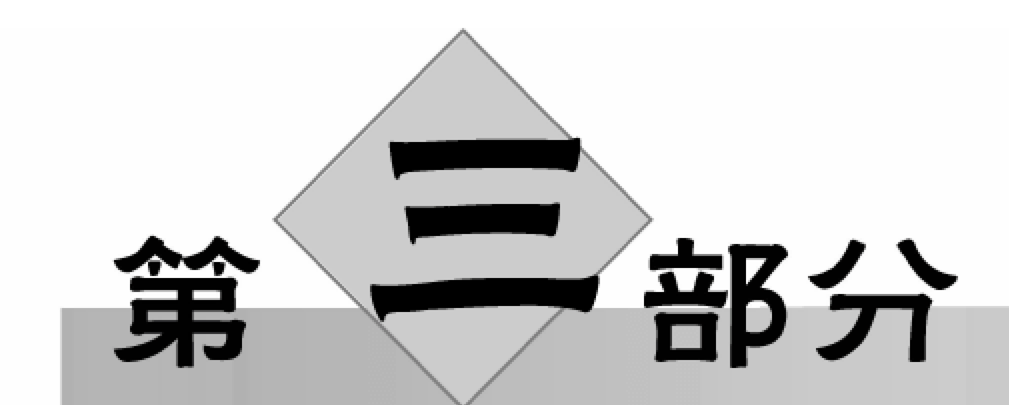
3. 为"佳衣屋"项目添加"删除"员工信息的功能。

要求:删除指定的员工信息。

提示:参考本章任务四完成

4. 本章学习的主要内容为:修改和删除数据表中的记录。请结合第 1 章课后练习的要求,完成项目中相应的修改或者删除数据表中记录的功能模块。





系统的安全设计



第 8 章　账号安全控制

本章学习要点：

- 熟练掌握 servlet 的生命周期；
- 熟练掌握 servlet 的配置；
- 熟练用 servlet 处理 cookie；
- 熟练掌握 servlet 会话追踪的功能；
- 熟练掌握 Filter 过滤器的编写和配置方法；
- 熟练掌握 Listener 监听器的编写和配置方法。

完成前 7 章的学习后，项目已经实现了许多功能模块，这些功能均涉及数据表增、删、改、查。但是这些功能都是对所有人开放的，这显然不符合系统的安全性要求。在第 2 章需求分析中已经明确：系统用户有三种角色，即管理员、总店店员和分店店员，三种角色应有不同的操作权限，本章将完善系统的账号安全控制。

8.1　任务一：用户登录功能的实现

任务目标：

用户输入用户名、密码，选择权限，如果上述信息都正确，则跳转到各角色下的默认主页，否则提示登录错误信息。

技能训练：

- Sevlet 的编写。
- Sevlet 的配置。

在本章中引入一种新的技术——Servlet 技术来完成这一任务。

8.1.1　Servlet 简介

要实现用户登录功能，首先要了解一种特殊的 Java 类——Servlet。与其他的 Java 相比，它的特殊点在于，Servlet必须继承Servlet类或者是它的子类，通常是HttpServlet

这个类。在 Web 应用中，Servlet 接受来自浏览器的请求，读取从客户端发送过来的数据，再根据请求类型和发送过来的数据去访问数据库或进行其他操作，并将结果返回给客户端。在下面的小节中采用 JSP+Servlet 完成用户登录功能。

8.1.2 MyEclipse 自动生成 Servlet

首先借助 MyEclipes 的向导功能完成一个 Servlet 的创建，如图 8-1～图 8-5 所示。

(1) 新建一个程序包

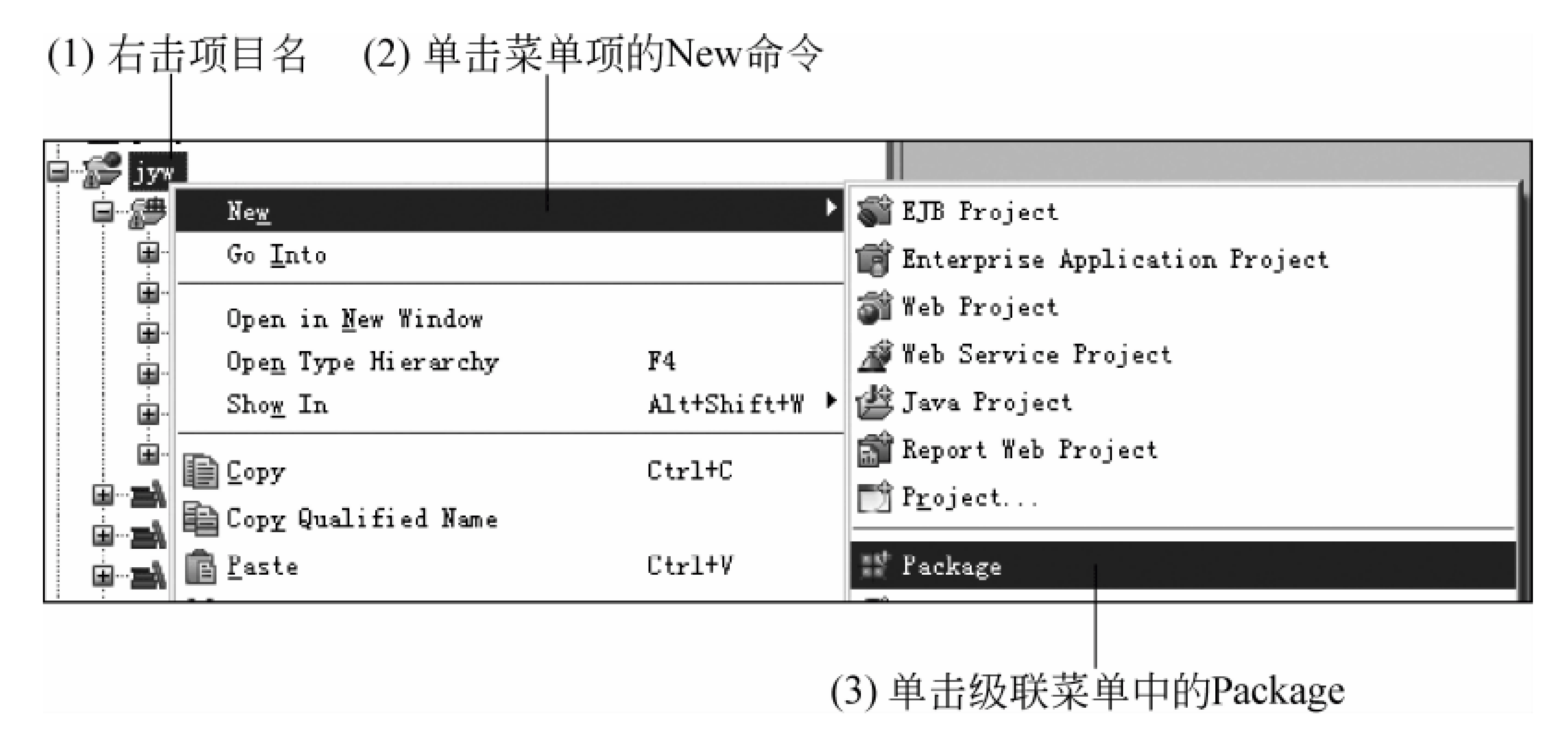


图 8-1 新建包

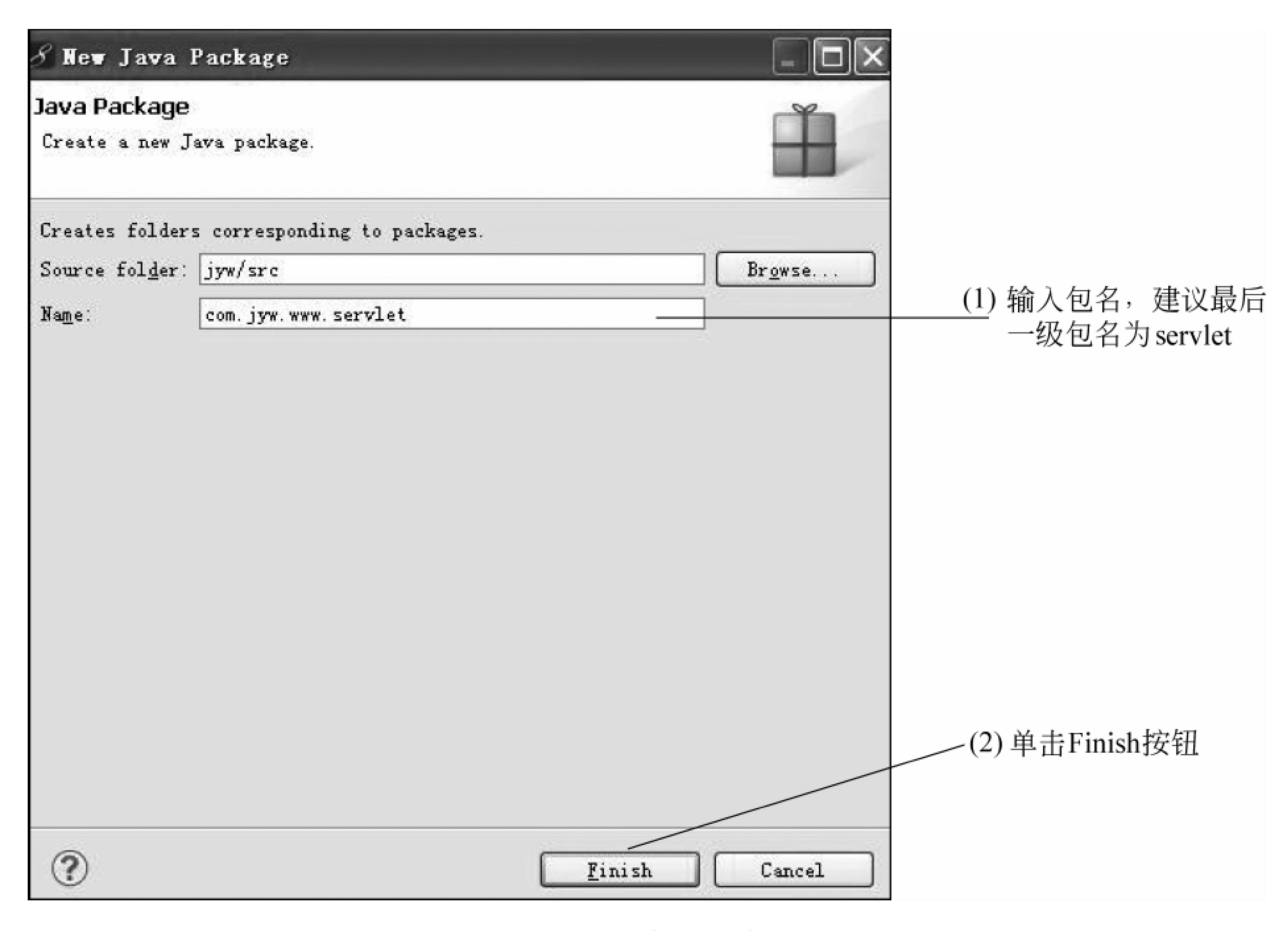


图 8-2 定义包名

（2）新建 Servlet



图 8-3　新建 servlet

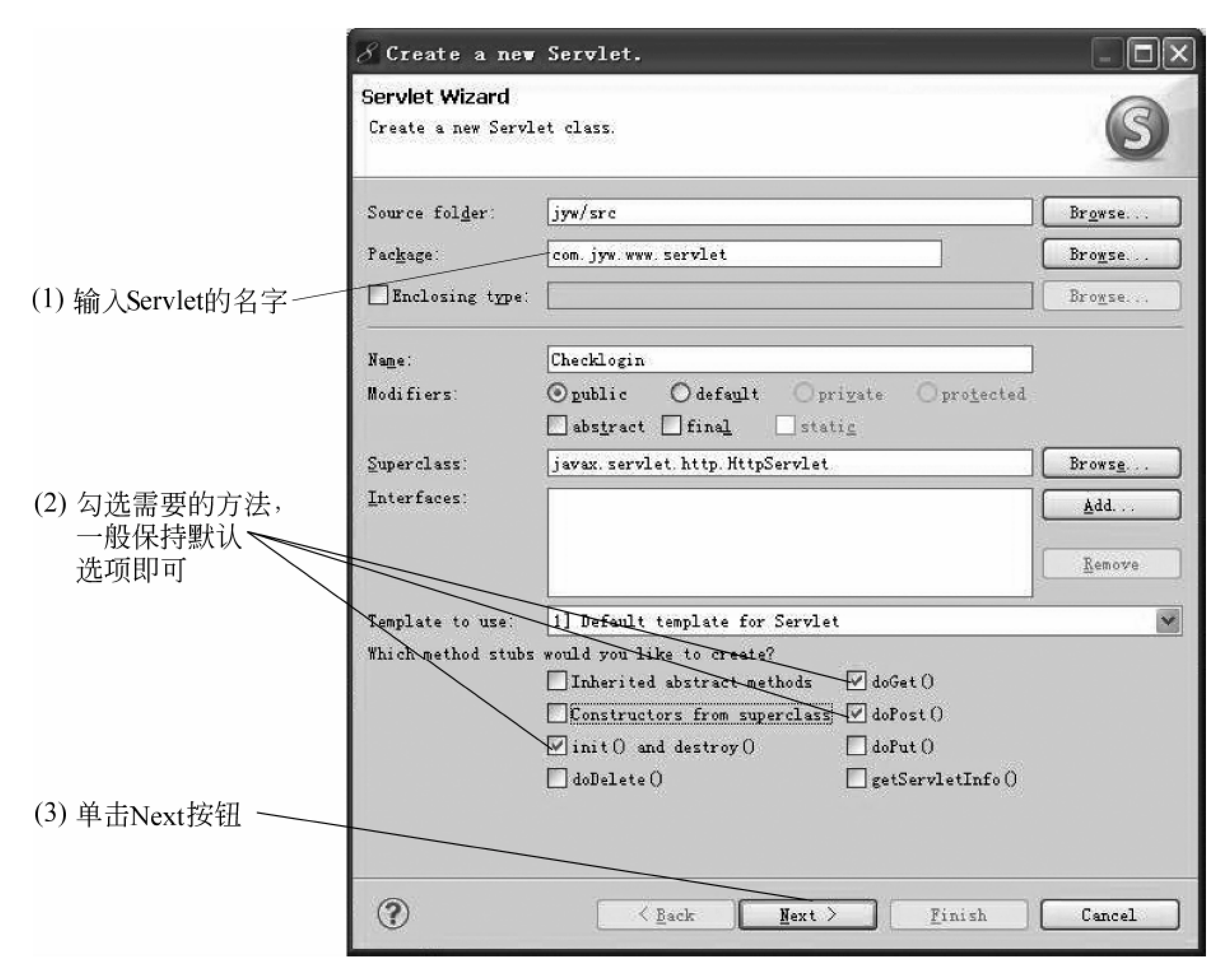


图 8-4　新建 Servlet 属性的设置

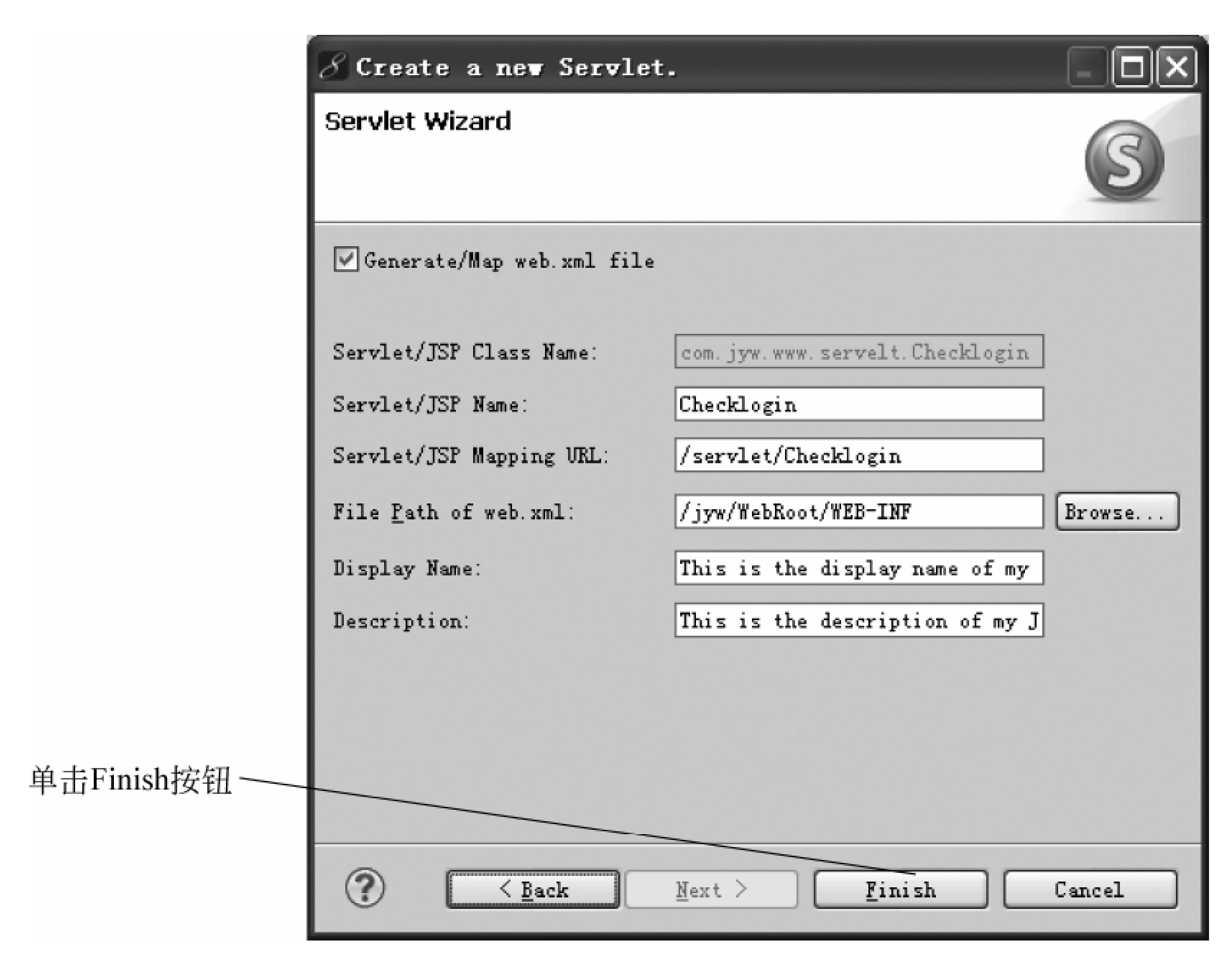


图 8-5 完成新建 servlet 的操作

完成向导引导的所有步骤后，MyEclipse 会自动生成如下的 CheckLogin.java 文件。

示例 8-1 CheckLogin.java 源代码

```
package com.jyw.www.servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Checklogin extends HttpServlet {

    /**
     * Destruction of the servlet. <br>
     */
    public void destroy() {
        super.destroy(); // Just puts "destroy" string in log
        // Put your code here
    }

    /**
     * The doGet method of the servlet. <br>
     *
     * This method is called when a form has its tag value method equals to get.
```

Servlet必须继承Servlet类或者是它的子类，通常是HttpServlet这个类

自动生成的destroy方法

```
 *
 * @param request the request send by the client to the server
 * @param response the response send by the server to the client
 * @throws ServletException if an error occurred
 * @throws IOException if an error occurred
 */                                   自动生成的doGet方法
public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
    out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01
        Transitional//EN\">");
    out.println("<HTML>");
    out.println("  <HEAD><TITLE>A Servlet</TITLE></HEAD>");
    out.println("  <BODY>");
    out.print("    This is ");
    out.print(this.getClass());
    out.println(", using the GET method");
    out.println("  </BODY>");
    out.println("</HTML>");
    out.flush();
    out.close();
}
/**
 * The doPost method of the servlet. <br>
 *
 * This method is called when a form has its tag value method equals to post.
 *
 * @param request the request send by the client to the server
 * @param response the response send by the server to the client
 * @throws ServletException if an error occurred
 * @throws IOException if an error occurred
 */                                            自动生成的doPost方法
public void doPost(HttpServletRequest request,HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
    out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01
        Transitional//EN\">");
    out.println("<HTML>");
    out.println("  <HEAD><TITLE>A Servlet</TITLE></HEAD>");
    out.println("  <BODY>");
    out.print("    This is ");
    out.print(this.getClass());
    out.println(", using the POST method");
    out.println("  </BODY>");
    out.println("</HTML>");
    out.flush();
    out.close();
```

```
    }

    /**
     * Initialization of the servlet. <br>
     *
     * @throws ServletException if an error occurs
     */
    public void init() throws ServletException {
        // Put your code here
    }

}
```

自动生成的init方法

MyElipse 自动生成的代码重写了 HttpServlet 中的 doPost()方法，使得它可以处理客户端的 Post 方式的请求。doPost()方法有两个参数，分别为 HttpServletRequest 和 HttpServletResponse，这两个参数分别用于表示客户端的请求和服务器端的响应。通过 HttpServletRequest，可以从客户端中获得很多客户端发送过来的信息，通过 HttpServletResponse 服务器可以向客户端发送信息。

8.1.3 编写 EmployeeDAO 类中的 isEmployee 方法

要完成本章任务一，需要验证客户端提交的账号、密码及权限是否与数据表 employee 中的记录一致，下面在 EmployeeDAO 中添加一个方法来完成判断，这种方法返回值为布尔型，方法命名规范一般以 is 开头，格式如下：

```
public Boolean isXxx(Xxx xxx)
```

按照命名规范为 EmployeeDAO 添加一个 isEmployee()方法，源代码示例如下。

示例 8-2 源代码 EmployeeDAO 类中的 isEmployee()方法

```
public Boolean isEmployee(Employee employee)
        throws Exception {
    InitialContext ctx=new InitialContext();
    DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
    Connection conn=ds.getConnection();
    PreparedStatement pstmt=conn.prepareStatement(
            "SELECT * FROM employee WHERE account=?and
        password=?and authority=?",
            ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY);
    pstmt.setString(1, employee.getAccount());
    pstmt.setString(2, employee.getPassword());
    pstmt.setInt(3, employee.getAuthority());
    ResultSet rs=pstmt.executeQuery();
    if (rs.first()) {
        return true;
    } else {
```

查找数据表employee中账号、密码和权限都与用户提交的值相同的记录

如果有满足条件的记录，则返回true

```
            return false;
        }

    }
```

8.1.4 改写 doPost 和 doGet 方法

对自动生成的 doPost()方法进行改写，首先要完成用户提交信息的接收，然后调用 8.1.3 小节中编写的 isEmployee()方法判断用户提交的信息是否正确，再根据情况作相应跳转，并使用 session 保存用户已登录的状态，如示例 8-3 所示。

示例 8-3　改写后的 doPost()方法

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
    request.setCharacterEncoding("GB18030");
    response.setContentType("text/html;charset=GB18030");
    EmployeeDAO employeeDao=new EmployeeDAO();
    Employee employee=new Employee();
    String account=request.getParameter("account");
    employee.setAccount(account);
    String password=request.getParameter("password");
    employee.setPassword(password);
    Integer authority=Integer.parseInt(request.getParameter("authority "));
    employee.setAuthority(authority);
    if (employeeDao.isEmployee(employee)) {
    switch (authority) {
        case 0:  request.getRequestDispatcher("/showSales.jsp").forward(request,
                 response);
                 request.getSession().setAttribute("adminAuth", account);
                 break; //管理员

        case 1:  request.getRequestDispatcher("/showOrders.jsp").forward(request,
                 response);
                 request.getSession().setAttribute("account", account);
                 break; // 总店店员
        case 2:  request.getRequestDispatcher("/searchStock.jsp").forward
                 (request, response);
                 request.getSession().setAttribute("account", account); // 分店店员
        }
    } else {
      request.getRequestDispatcher("/error.jsp").forward(request,response);
    }
}
```


为了避免用户直接跳过登录页面而在浏览器中输入地址加参数的形式来登录，可以将 doGet()改写为示例 8-4 所示的处理方式。

示例 8-4　改写后的 doGet()方法

```
public void doGet(HttpServletRequest request,HttpServletResponse response)
                    throws ServletException,IOException
{
    PrintWriter out=response.getWriter();
    out.println("请从登录页面登录");
}
```

8.1.5　配置 servlet 映射

Servlet 编写完成后，为了能够从 JSP 中使用 Servlet，还必须在项目的 Web-INF 目录下的 Web. xml 中对它进行配置。对 Servlet LoginCheck 的配置，是在 Web. xml 的<Webapp></Webapp>标签之间插入示例 8-5 所示的子标签。

示例 8-5　servlet 的配置

```
<servlet>
      <servlet-name>LoginServlet</servlet-name>
      <servlet-class>com.jyw.www.servlet.LoginCheck </servlet-class>
    </servlet>
    <servlet-mapping>
      <servlet-name>LoginServlet </servlet-name>
      <url-pattern>/checklogin</url-pattern>
  </servlet-mapping>
```

servlet别名不能含空格。配置别名，可以避免将servlet的实际路径暴露给客户端的用户

指明有此别名的Servlet类

JSP中访问此servlet的路径

8.1.6　编写登录页 login. jsp

Servlet 编写配置完成后，就可以在 JSP 文件中访问它了。示例 8-6 是"佳衣屋"系统的登录页面的源代码，注意表单的 action 属性使用了 servlet 访问路径。

示例 8-6　源代码 login. jsp

```
<%@page language="java" import="java.util.*"
        contentType="text/html;charset=gb18030" pageEncoding="gb18030"%>
<center>
        <form name="form1" method="post" action="/checklogin">
                用户名
                <input type="text" name="account" size="15">
                <br>
                密 码
                <input type="password" name="password" size="15">
                <br>
                选择身份
                <select name="authority">
                    <option value="0">
```

Web.xml中配置的Servlet LoginCheck的访问路径

```
                    管理员
                </option>
                <option value="1">
                    总店店员
                </option>
                <option value="2">
                    分店店员
                </option>
            </select>
            <br>
            <input type="submit" value="登录" />
        </form>
</center>
```

登录页的执行效果如图 8-6 所示。

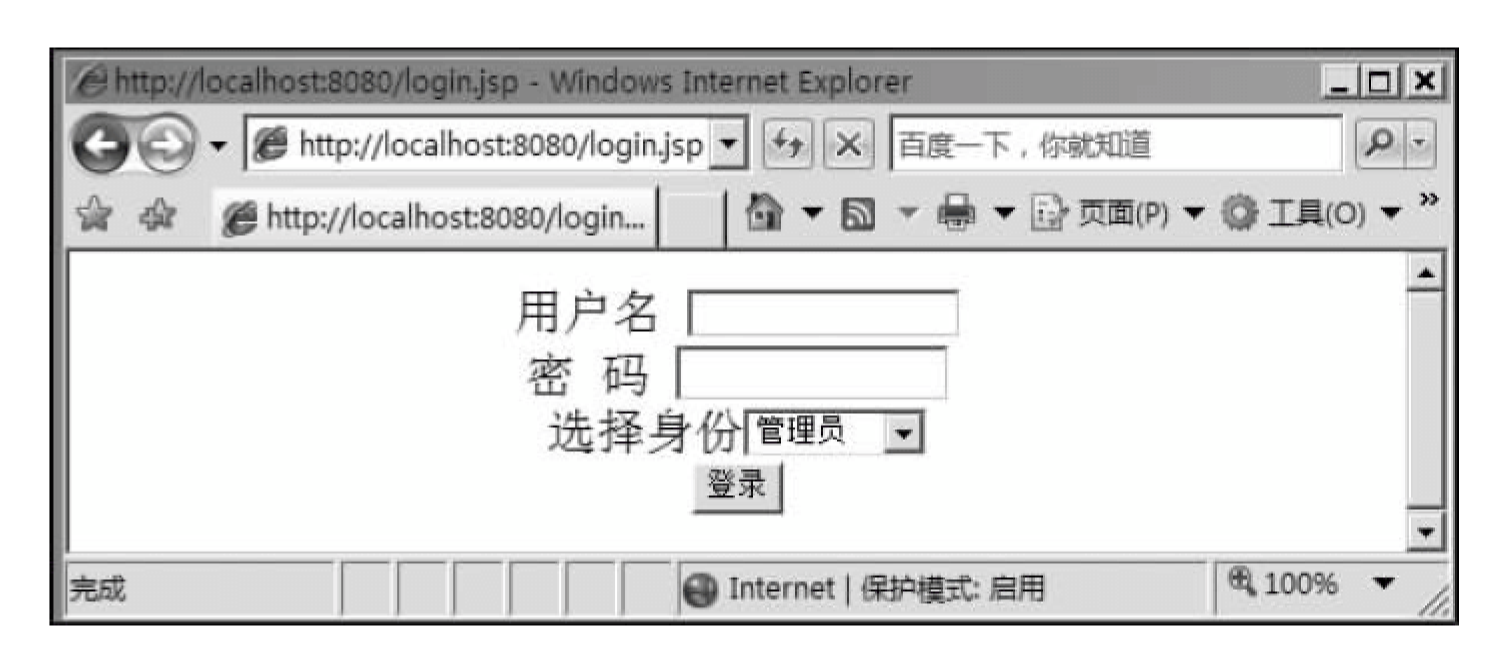


图 8-6　登录页的执行效果

8.2　任务二：验证码的生成

问题：我们在登录一些 Web 应用，如网银、电子邮件等系统时，系统除了要求输入用户名和密码外，还往往要求输入一个“验证码”，这是为什么？又如何在我们的系统中实现这一功能呢？

许多黑客工具使用穷举法来破解用户的密码，输入验证码是为了防止有恶意的用户使用自动注册机等工具来不停地进行登录尝试。验证码，就是在每次访问登录界面时，随机生成一串新的字母、数字甚至中文，一方面将这个验证码生成图片显示给用户；另一方面将它保存在服务器上，当用户进行登录的时候，输入验证码，通过比较用户输入的验证码和保存在服务器上的验证码是否一致，判断用户是否在恶意使用登录功能。使用图片文字显示验证码主要是为了防止有些工具能够通过分析网页而得到验证码。

8.2.1　编写生成验证码图片的 servlet

本小节中使用 servlet 来产生随机的四位数验证码，并且将这四位验证码以图像的形

式输出到客户端。

示例 8-7 产生图像验证码的 CreatValidationCode.java

```
package com.jyw.www.servlet;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Random;
import java.awt.*;
import java.awt.image.*;
import javax.imageio.*;

public class CreatValidationCode extends HttpServlet
{
    public void doPost(HttpServletRequest request,
            HttpServletResponse response)
                throws ServletException,IOException
    {                                                        设置输出内容的MIME格式
        response.setContentType("image/jpeg");
        response.setHeader("Pragma","No-cache");             设置页面不缓存，
        response.setHeader("Cache-Control","no-cache");      避免在浏览器端
        response.setDateHeader("Expires",0);                 使用缓存里的图像

        OutputStream out=response.getOutputStream();
        int width=80, height=20;
        BufferedImage image=new BufferedImage(width, height,
            BufferedImage.TYPE_INT_RGB);                     设置缓冲图片的宽和高

        Graphics g=image.getGraphics();
                                                             得到Graphics对象，以绘制图像

        Random random=new Random();                          生成随机类
        g.fillRect(0, 0, width, height);
        //                                                   设置字体
        g.setFont(new Font("Times New Roman",Font.ITALIC,18));
        String sRand="";
        for (int i=0;i<4;i++){                               产生一个随机数
            String rand=String.valueOf(random.nextInt(10));
            sRand+=rand;
                                                             设置随机颜色
            g.setColor(new Color(20+random.nextInt(110),20
                +random.nextInt(110),20+random.nextInt(110)));

            g.drawString(rand,20*i+6,16);                    绘制图片
        }
        g.dispose();

        //
```

```
        ImageIO.write(image, "JPEG", out);      将缓冲图片输出到页面
    }
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
                throws ServletException,IOException
    {
        doPost(request,response);
    }                                          Get方式的请求处理与Post方式相同
}
```

8.2.2　验证码图片 servlet 的配置

在 Web.xml 的<Web-apps>中插入下列代码。

```
<servlet>
    <servlet-name>CreatValidationCode</servlet-name>
    <servlet-class>com.jyw.www.servlet.CreatValidationCode</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>CreatValidationCode</servlet-name>
    <url-pattern>/creatValidationCode</url-pattern>
</servlet-mapping>
```

8.2.3　使用验证码图片生成的 servlet

下面改写示例 8-6，添加图片验证码到登录表单中。

示例 8-8　添加了图片验证码后的 login.jsp

```
<%@page language="java" import="java.util.*"
    contentType="text/html;charset=gb18030" pageEncoding="gb18030"%>
<center>
    <form name="form1" method="post" action="/checklogin">
        用户名
        <input type="text" name="account" size="15">
        <br>
        密 码
        <input type="password" name="password" size="15">
        <br>
        验证码
        <input TYPE="text" size="10" name="valCode">
        <img src="/creatValidationCode">
        <br>
        选择身份          使用验证码生成servlet
        <select name="authority">
            <option value="0">
```

```
                管理员
            </option>
            <option value="1">
                总店店员
            </option>
            <option value="2">
                分店店员
            </option>
        </select>
        <br>
        <input type="submit" value="登录" />
    </form>
</center>
```

在这个 HTML 程序中，定义了一个表单，用于接收用户的登录名和密码，另外，在这两个表单元素的基础上，加上了一个验证码的输入框，用户必须在登录的时候输入这个验证码，如图 8-7 所示。

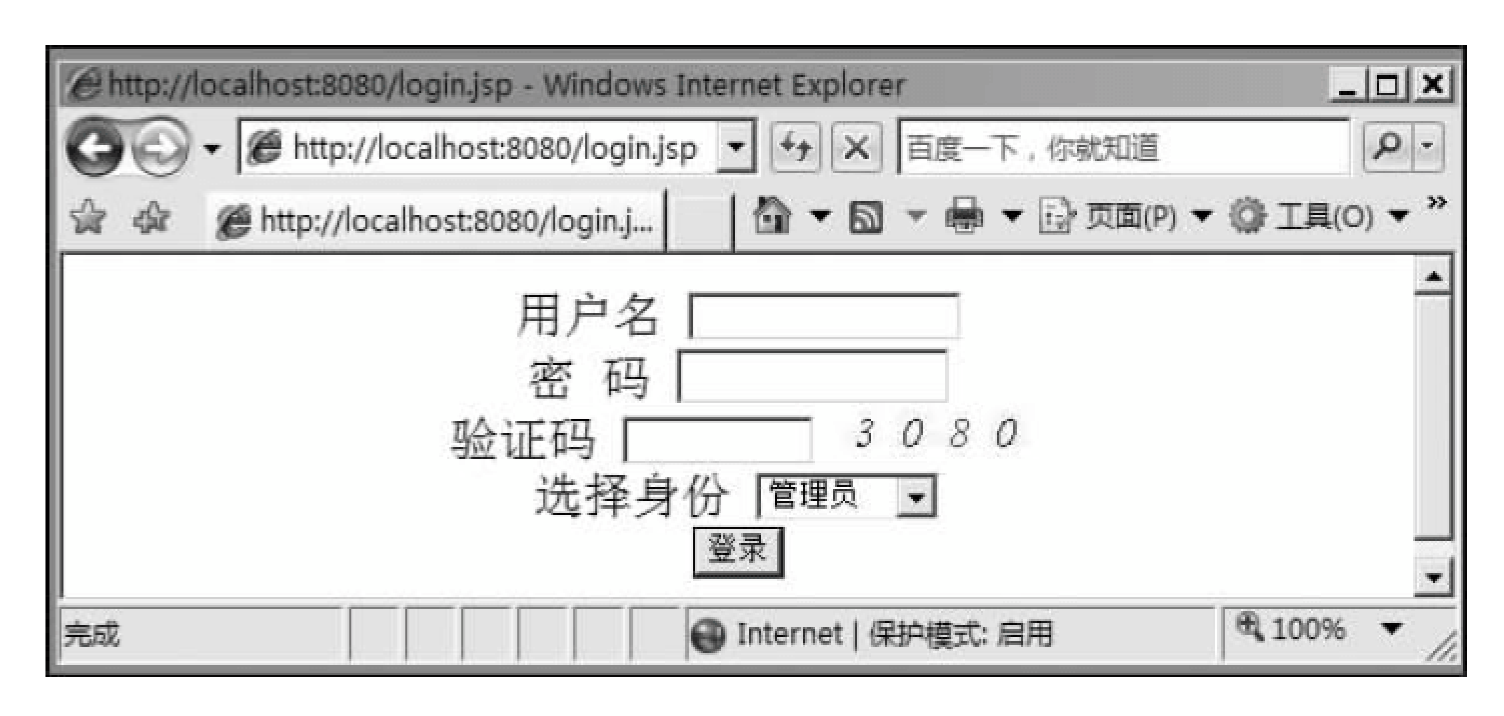


图 8-7　带验证码的登录页

8.2.4　校对验证码

下面进一步修改示例 8-3，加入对验证码的校对。

示例 8-9　添加验证码校对功能的 doPost()方法

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    request.setCharacterEncoding("GB18030");
    response.setContentType("text/html;charset=GB18030");
    EmployeeDAO employeeDao=new EmployeeDAO();
    Employee employee=new Employee();
    String account=request.getParameter("account");
    employee.setAccount(account);
    String password=request.getParameter("password");
    employee.setPassword(password);
    Integer previlege=Integer.parseInt(request.getParameter("authority"));
    employee.setAuthority (previlege);
```

```
    String code= (String) sess.getAttribute("code");        获取生成的验证码
    String valCode=request.getParameter("valCode");         获取用户输入的验证码
    try {
      if (valCode !=null && valCode.equals(code)) {          验证码的校对
        if (employeeDao.isEmployee(employee)) {
           switch (previlege) {
               case 0: request.getRequestDispatcher("/showSales.jsp").forward
                       (request, response);
                       break; // 管理员
               case 1: request.getRequestDispatcher("/showOrders.jsp").forward
                       (request, response);
                       break; // 总店店员
               case 2: request.getRequestDispatcher("/searchStock.jsp").forward
                       (request, response); // 分店店员
           }
        }else {
         request.getRequestDispatcher("/error.jsp").forward(request,response);
        }
      }else{
        PrintWriter out=response.getWriter();
        out.print("验证码错误");
        out.print("<a href='/login.jsp'>"+"回登录页"+"</a>");
        out.flush();
        out.close();
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
}
```

修改完成后,就可以顺利实现验证码功能了。

8.3　任务三:用户名和密码在客户端的保存

问题:打开“淘宝”登录页,在输入密码的右下方有一个“十天免登录”选项,如果勾选了这个选项,十天之内,在同一台 PC 上打开淘宝网站,会发现用户已经自动登录了,这个功能又是如何实现的呢?

这个功能是通过使用 Cookie,在 PC 上保留用户输入的账号和密码一段时间来实现的。

任务目标:

在登录页添加用户选择是否保存用户名和密码,并且选择保存多久的功能。

技能训练:

- Cookie 的使用。
- 在 Servlet 中处理 Cookie。

8.3.1 Cookie 基础

网站个人化是 Cookie 最有益的用途之一。Cookie 浏览某网站时，网站存储在浏览器所在的客户端上的一个小文本文件，它记录了用户 ID，密码、浏览过的网页、停留的时间等信息，当同一主机再次来到该网站时，网站通过读取 Cookie，得知该主机的相关信息，就可以做出相应的动作，比如不用输入用户名、密码就直接登录等。

用户使用某台主机第一次登录网站，输入用户名和密码后，用户名和密码通过 Cookie 保存在该主机上，当用户下一次通过这台主机访问这个网站的时候，可以直接从本地 Cookie 中读出用户名和密码，不用再重新输入了。

因为需要将信息保存在客户端的机器上，所以，Cookie 可能会带来安全性问题。因此，项目开发过程中，不要将敏感的信息(如银行账号等)保存到 Cookie 中，或者在保存这些信息之前提示用户，确认信息不是保存在公共计算机上。

处于安全性考虑，浏览器可以被设置成拒绝 Cookie，所以项目的功能实现不要对 Cookie 有很大的依赖性。

8.3.2 编写处理 Cookie 的类

Cookie 的实例，可以通过 Cookie 的构造器来创建，它接收两个 String 类型参数，用于指定 Cookie 的属性名称和属性值。

在创建好 Cookie 对象后，需要将它发送到客户端，可以使用 HttpServletResponse 的 addCookie()方法来实现，它接收一个 Cookie 类型的值，并将这个 Cookie 发送到客户端。可以多次使用 addCookie()方法来将不同的 Cookie 发送到客户端。

可以通过对象 Cookie 的 setMaxAge()方法来设置 cookie 的"生存期"，这个方法接收一个以秒为单位的整数参数，表示这个 cookie 可以在客户端的保存时间。如果将它设置成 0，则表示在客户端删除了这个 Cookie。

示例 8-9 按照上述步骤处理 Cookie 的源代码，在这个程序中，首先创建了两个 Cookie 对象，分别用来储存表单中传递过来的用户名和密码，然后根据客户端传递的 saveCookie 参数，决定是向客户端发送 cookie，还是删除以前存储的 cookie，通过 Cookie 对象的 setMaxAge()可以实现这个目的。

示例 8-10 Cookie 处理类 CookieUtil. java

```
package com.jyw.www.util;

import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class CookieUtil {
```

```
    public static void addCookies(HttpServletRequest request,
    HttpServletResponse response){
        String account=request.getParameter("account");      } 获取账号和密码
        String password=request.getParameter("password");    }
        Cookie userCookie=new Cookie("account", account);    } 创建Cookie
        Cookie pwdCookie=new Cookie("pwd", password);        } 的新实例
        if (request.getParameter("saveCookie") !=null
                && request.getParameter("saveCookie").equals("Yes")) {
                                              如果用户要求记住用户名和密码
            userCookie.setMaxAge(7 * 24 * 60 * 60);   } 设置Cookie生存期为7天
            pwdCookie.setMaxAge(7 * 24 * 60 * 60);    }
        } else {
            userCookie.setMaxAge(0);   } 如果用户没有要求记住用户名
            pwdCookie.setMaxAge(0);    } 和密码，则删除Cookie
        }
        response.addCookie(userCookie);   } 将Cookie发送到客户端
        response.addCookie(pwdCookie);    }
    }
}
```

8.3.3　在 servlet 中调用 Cookie 处理类

用户输入的用户名和密码是被 Servlet 接收并进行下一步处理的，所以需要继续改写示例 8-8 中的 doPost 方法，在其中调用 8.3.2 小节编写的 Cookie 处理类 CookieUtil.java。

示例 8-11　LoginServlet.java

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    request.setCharacterEncoding("GB18030");
    response.setContentType("text/html;charset=GB18030");

    CookieUtil.addCookies(request, response);
                          调用实例8-10中的Cookie处理方法，按
                          照用户要求，保存用户的账号和密码

    EmployeeDAO employeeDao=new EmployeeDAO();
    HttpSession sess=request.getSession(true);
    Employee employee=new Employee();
    String account=request.getParameter("account");
    employee.setAccount(account);
    String password=request.getParameter("password");
    employee.setPassword(password);
    Integer authority=Integer.parseInt(request.getParameter("authority"));
    employee.setAuthority(authority);
```

```
        String code=(String) sess.getAttribute("code");
        String valCode=request.getParameter("valCode");
        try {
          if (valCode !=null && valCode.equals(code)) {
            if (employeeDao.isEmployee(employee)) {
              switch (authority) {
                case 0:
                  request.getRequestDispatcher("/showSales.jsp").forward
                  (request, response);break;
                case 1:
                  request.getRequestDispatcher("/showOrders.jsp").forward
                  (request, rsponse);break;
                case 2:
                  request.getRequestDispatcher("/searchStock.jsp").forward
                      (request,response);
              }
            } else {
                  request.getRequestDispatcher("/error.jsp").forward(request,
                        response);
            }
          }else{
                  PrintWriter out=response.getWriter();
                  out.print("验证码错误");
                  out.print("<a href='/login.jsp'>"+"回登录页"+"</a>");
                  out.flush();
                  out.close();
          }
        } catch (Exception e) {
                  e.printStackTrace();
        }
    }
```

8.3.4 改写登录页面

在将 Cookie 发送到客户端以后,可以使用 HttpServletRequest 的 getCookies()方法从客户端获得这个网站所有的 Cookie,它返回包含所有本站 Cookie 的数组,然后通过遍历这个数组就可以获得所需的 Cookie。

在 login.jsp 中加入上述处理,并且添加一个列表菜单,让用户可以选择是否记住用户名和密码。

示例 8-12 login.jsp

```
<%@page contentType="text/html;charset=gb18030" language="java" import="java.
util.*"%>
<html>
  <head><title>登录页</title></head>
```

```
  <body align="center">
      <jsp:include page="menu.jsp"/>
<%
    Cookie[] mycookies=null;
    int i;
    String account=null;
    String password=null;
    if (request.getCookies() !=null) {
      mycookies=request.getCookies();         取出网站所有的Cookie
      for (i=0; i<mycookies.length; i++) {    遍历所有的Cookie
        if (mycookies[i].getName().equals("account")) {
          account=mycookies[i].getValue();    找出存储在Cookie中的账号

        }else if (mycookies[i].getName().equals("pwd")) {
          password=mycookies[i].getValue();   找出存储在Cookie中的密码
        }
      }
    }
%>

<form name="form1" method="post" action="/jyw/checklogin">
      用户名<INPUT type="text" name="username"
              value='<%=(account!=null)?account:""%>'>
                                  将账号显示在用于
                                  输入用户名的元素中

密　码<input type="password" name="password"
                  value='<%=(password!=null)?password:""%>'>
请选择身份:<select name="authority">
  <option value="0">管理员</option>          将密码显示在用于
  <option value="1">总店店员</option>        输入密码的元素中
  <option value="2">分店店员</option>
</select>
选择　<select name="saveCookie">
      <option value="No">不记账号和密码</option>
      <option value="Yes">记住账号和密码</option>
      </select>
  <input type="submit" value="登录"/>
  </p>
</form>
<body>
<html>
```

修改后的登录页执行效果如图 8-8 所示，记住用户名和密码的任务就完成了。



图 8-8　可以记住用户账号和密码的登录页

8.4　任务四：用户密码的 MD5 加密

各分店的店员应能根据所分配的账号登录系统进行商品库存或销售信息的查询、更改等操作。这就需要事先为员工分配账号等信息，并设置初始密码。这个功能可以用第 5 章学到的数据表记录的插入技术来完成。但是本章对这一功能的实现提出了新的要求：为了安全起见，需要为用户密码加密。

任务目标：

添加各店员账号的初始化录入功能。

要求：

- 要求输入店员的账号、密码、真实姓名，选择其所属分店，单击“提交”按钮后，存储到员工信息表中。
- 对用户密码进行加密存储，即在数据表中看到的用户密码是经过加密处理后的密文。

技能训练：

- 熟练掌握向数据表添加一条信息的实现技术。
- MD5 加密算法的编写。

8.4.1　店员基本信息录入页面的编写

示例 8-13 和示例 8-14 按照任务要求给出员工信息录入功能的前台处理页，没有使用新的技能，读者可以当作巩固性的练习来完成编写。

示例 8-13　employeeReg.jsp 员工信息录入表单。

```
<%@page language="java" import="java.util.*" contentType="text/html;
charset=gb18030"%><center>
    填写注册信息
    <form name="form1" method="post" action="addEmployee.jsp">
        用户名
        <input type="text" name="account">
        <br>
        密 码
        <input type="password" name="password">
        <br>
        姓 名
        <input type="text" name="name">
        <br>
        所属分店
        <select name="storenum">
            <option value="0">
                总店
            </option>
            <option value="1">
                一分店
            </option>
            <option value="2">
                二分店
            </option>
        </select>
        <br>
        授 权
        <select name="authority">
            <option value="0">
                管理员
            </option>
            <option value="1">
                总店店员
            </option>
            <option value="2">
                分店店员
            </option>
        </select>
        <br>
        <input type="submit" name="submit" value="添加" />
        <input type="reset" name="reset" value="重置" />
    </form>
</center>
```

执行效果如图 8-9 所示。

示例 8-14　addEmployee.jsp

```
<%@page language="java" import="java.util.*,com.jyw.www.dao.*"
    contentType="text/html;charset=gb18030"%>
```



图 8-9　员工信息录入页

```
<%  request.setCharacterEncoding("GB18030");
    response.setContentType("text/html;charset=GB18030");
%>
<jsp:useBean id="employee" class="com.jyw.www.entity.Employee" scope="page"/>
  <jsp:setProperty name="employee" property="*"  />
<%
EmployeeDAO eDao=new EmployeeDAO();
eDao.save(employee);

%>
```

8.4.2　编写对密码进行 MD5 加密的类

用户密码以明文的方式存储在数据表中是不安全的，一般都需要经过加密后再存储。MD5 算法是一种不可逆的数据加密算法，即在获得密文的情况下也无法还原出明文，有较好的安全性，被许多 Web 应用所采用。下面编写一个用于 MD5 加密的类和方法，将其放在项目的 util 包中。

示例 8-15　MD5Hashing 类中 getMD5Hashing 方法的编写

```
package com.jyw.www.util;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class MD5Hashing
{
    public String getMD5Hashing(String password) throws NoSuchAlgorithmException
    {
        MessageDigest md;
        md=MessageDigest.getInstance("MD5");
        md.update(password.getBytes());

         byte byteData[]=md.digest();
```

```
        StringBuffer sb=new StringBuffer();
        for (int i=0; i<byteData.length; i++) {
          sb.append(Integer.toString((byteData[i]&0xff)+0x100,16).substring(1));
        }

        StringBuffer hexString=new StringBuffer();
        for (int i=0;i<byteData.length;i++) {
            String hex=Integer.toHexString(0xff & byteData[i]);
            if(hex.length()==1) hexString.append('0');
            hexString.append(hex);
        }
        return  hexString.toString();

    }

}
```

8.4.3 编写 EmployeeDAO 类中的 save 方法

DAO 类中 save()方法的编写规则，在第 5 章中已经给出了示例，完成此任务的不同之处在于在 save()方法中需要调用示例 8-15 中的 getMD5Hashing 方法，获得密文返回值后，将密文写入相应数据表中。

示例 8-16 EmployeeDAO 类中 save()方法

```
public void save(Employee employee) {
    try {
        InitialContext ctx=new InitialContext();
        DataSource ds= (DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
        Connection conn=ds.getConnection();
        PreparedStatement pstmt=conn.prepareStatement(
                "INSERT INTO employee
            (account,password,storenum,name,previlige) VALUES (?,?,?,?,?)",
                ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY);
        pstmt.setString(1, employee.getAccount());
        String password=employee.getPassword();
        MD5Hashing mh=new MD5Hashing();
        pstmt.setString(2, mh.getMD5Hashing(password));
        pstmt.setInt(3, employee.getStorenum());
        pstmt.setString(4, employee.getName());
        pstmt.setInt(5, employee.getAuthority());
        pstmt.executeUpdate();
        pstmt.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

将加密后的密码写入数据表中

录入新的员工信息后，打开 employee 数据表，可以看到在 password 列中的密码都是以密文的方式存储的，看不出真实的密码，如图 8-10 所示。

用户输入的密码在数据库中以密文的形式存储

	id	account	password	storenum	name	authority
□	2	lim	5f4dcc3b5a	0	李明	0
□	5	chenf	e10adc3949ba59abbe56e057f20f883e	1	陈芳	0
□	6	liuh	e10adc3949ba59abbe56e057f20f883e	0	刘华	1
□	7	wangt	e10adc3949ba59abbe56e057f20f883e	1	王婷	2
*	(Auto)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

数据库：jyw 表单：employee

图 8-10　经 md5 加密后对用户密码的示意

8.4.4　改写 EmployeeDAO 类中的 isEmployee　方法

完成 8.4.3 小节后，用户在登录时，即便输入的账号和密码正确也不能正常登录，这是因为用户登录时输入的是明文的密码，而数据表中的密码以密文的形式存储。要恢复登录功能就需要把用户输入的密码以同样的算法加密，在与数据表中的密文相比较来判断输入的密码是否正确。所以，本章任务一中示例 8-2 的 isEmployee()方法要进行修改如下。

示例 8-17　EmployeeDAO 类中的 isEmployee()方法

```
public Boolean isEmployee(Employee employee) throws Exception {
    InitialContext ctx=new InitialContext();
    DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/mysql");
    Connection conn=ds.getConnection();
    PreparedStatement pstmt=conn.prepareStatement(
                "SELECT * FROM employee WHERE account=? and
            password=? and authority=?",
                ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY);
    pstmt.setString(1, employee.getAccount());
    String password=employee.getPassword();
    MD5Hashing mh=new MD5Hashing();
    pstmt.setString(2, mh.getMD5Hashing(password));
    pstmt.setInt(3, employee.getAuthority());
    ResultSet rs=pstmt.executeQuery();
    if (rs.first()) {
        return true;
    } else {
        return false;
    }
}
```

以加密后的密码作为查询条件

修改完成后，登录功能就可以恢复了。

8.5　任务五：已登录用户的身份跟踪

问题：用户一次登录 Web 应用系统后，无论切换到哪个页面，都可以被识别；某些 Web 应用系统，如淘宝网，还会将用户名一直显示在"页眉"处；当用户离开计算机较长时间后，发现已经自动退出了系统。这是如何实现的呢？

任务目标：

用户登录"佳衣屋"系统后，在任何一个网页均可识别用户身份，显示"欢迎×××"的信息，并给出"退出"超链接，供用户使用完系统功能后，退出系统之用。用户单击"退出"超链接后，欢迎信息和"退出"超链接不再出现。

技能训练：

- Servlet 处理会话追踪。
- Session 常用方法的使用。

8.5.1　Session 简介

本节中，采用 Session(会话)技术来完成任务五。

Session 的基本思路是：给每个客户端分配一个不重复的 ID 来区分不同的客户端，而将对应各个客户需要保存的数据保存在服务器的内存中，因为不需要写到客户端的文本中，所以，在服务器中可以存放任何的对象而不单纯是 String 数据。

Session 封装在 javax.servlet.http.HttpSession 这个接口中，这个接口构建在 Cookie 或者 URL 重写的基础上。要得到一个 HttpSession 的实例，可以通过 HttpServletRequest 的 getSession()方法来获得，HttpServletRequest 有两个重载的 getSession()方法，一个不带任何参数，一个接收 boolean 类型的值。getSession()方法和 getSession(true)功能一样，如果对应的客户端已经产生过一个 session，那么就会返回这个旧的 session，否则，这个方法将产生一个 session ID 并且和对应的客户端绑定在一起。而如果 getSession(false)表示对应的客户端已经有 session，那么返回这个旧的 session，否则，不会产生新的 session。可以使用 HttpSession 对象上的 isNew()方法来判定这个 session 是否为新建的。

使用 Session 技术完成任务五的要点有以下四点。

(1) 已登录用户的用户名需要保存在 Session 中。使用 Session 对象的 void setAttribute(String attrName, Object obj)方法。

(2) 设置 Session 处于连续非活跃状态的生存期。使用 Session 对象的 void setMaxInactiveInterval(Integer seconds)方法。

(3) 在需要使用用户名的地方，从 Session 中取出。使用 Session 对象的 Object getAttribute(String attrName)方法。

(4) 用户单击“退出”超链接后使 Session 失效。使用 Session 对象的 void invalidate() 方法。

8.5.2 Servlet 中使用 Session

首先来改写 LoginServlet.java 中的 doPost 方法,将用户已登录的状态信息保存到 Session 中。

示例 8-18 LoginServlet.java

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    request.setCharacterEncoding("GB18030");
    response.setContentType("text/html;charset=GB18030");
    CookieUtil.addCookies(request, response);
    EmployeeDAO employeeDao=new EmployeeDAO();
    HttpSession sess=request.getSession(true);
    Employee employee=new Employee();
    String account=request.getParameter("account");
    employee.setAccount(account);
    String password=request.getParameter("password");
    employee.setPassword(password);
    Integer authority=Integer.parseInt(request.getParameter("authority"));
    employee.setAuthority(authority);
    String code= (String) sess.getAttribute("code");
    String valCode=request.getParameter("valCode"); // 获取用户输入的验证码
    try {
          if (valCode !=null && valCode.equals(code)) {
            if (employeeDao.isEmployee(employee)) {
              request.getSession().setAttribute("account", account);
                                        把用户登录时输入的用户名保存在Session中
              request.getSession().setAttribute("authority", authority);
                                        把用户登录时选择的角色保存在Session中
              request.getSession().setMaxInactiveInterval(10 * 60);
              switch (authority) {             设置Session非活跃状态的生存期为10分钟
                case 0:
                  request.getRequestDispatcher("/showSales.jsp").forward(
                      request, response);
                  break;
                case 1:
                  request.getRequestDispatcher("/showOrders.jsp")
                      .forward(request, response);
                break;
                case 2:
                  request.getRequestDispatcher("/searchStock.jsp").forward(
                  request, response);
              }
          } else {
```

```
                request.getRequestDispatcher("/error.jsp").forward(request,
                response);
              }
            }else{
              PrintWriter out=response.getWriter();
              out.print("验证码错误");
              out.print("<a href='/login.jsp'>"+"回登录页"+"</a>");
              out.flush();
              out.close();
            }
        } catch (Exception e) {
              e.printStackTrace();
        }
}
```

8.5.3　编写显示欢迎信息页眉的 JSP 页面

本小节编写一个简单 header 页面来示例如何从 Session 中取出一个属性的值。

示例 8-19　header.jsp

```
<%
    String account=(String) session.getAttribute("account");
    if (account !=null) {
        out.print(account);
        %>已登录
        <a href="logout.jsp">退出</a>
        <%
    }
%>
```

从Session中取出用户账号对象，并强制转换为字符串

用户 chenf 登录后的执行效果引用了 header.jsp 的页面执行效果，如图 8-11 所示。



图 8-11　带欢迎信息和“退出”超链接的页面

8.5.4 编写实现退出系统的 JSP 页

由于 Session 中“存储”了用户登录的信息，所以如果使 Session“失效”，则用户已登录的信息就不复存在了，按照这一思路，编写一个简单的退出页面来示例如何使 Session 失效。

示例 8-20 logout.jsp

```
<%
session.invalidate();          使Session失效，完成退出
response.sendRedirect("login.jsp");
%>
```

用户单击 header.jsp 中的“退出”超链接，将调用示例 8-19 的 logout.jsp，使 Session 失效。

8.6 任务六：角色权限的过滤

到目前为止，为“佳衣屋”系统添加了许多功能模块，这些功能无论用户是否登录都可以使用，甚至添加系统用户这类权限的管理员才有权操作的功能也是开放的，这显然不符合安全性的要求，任务六就要求为“佳衣屋”系统按角色分配操作权限。

任务目标：

按照指定的系统结构图，对系统用户按角色进行权限划分，例如：只有管理员登录后才能实现员工信息录入的功能。

要求：

- 将受控文件放到相应的目录中。
- 参考练习 10.3 可以为项目添加管理员登录 filter 过滤器。
- 正确配置 Web.xml 文件。

技能训练：

- Filter 的编写。
- Filter 的配置。

8.6.1 Filter 过滤器简介

过滤器是一种小型的、可插入的 Web 组件，其提供了对 Web 应用程序的前期处理和后期处理的逻辑控制，可以拦截请求和响应，以便查看、提取或以某种方式操作正在客户端和服务器之间进行交换的数据。与 Servlet 类似，过滤器也需要在 Web 应用配置文件

(即 Web. xml)中进行配置部署。

过滤器并不是 Servlet,过滤器并不实际创建一个请求。而是请求到达一个 Servlet 前的预处理程序或响应离开 Servlet 后的后处理程序。一个过滤器能够实现如下功能:

- 在一个 Servlet 被调用前截获该调用。
- 在一个 Servlet 被调用前检查请求。
- 修改在实际请求中提供了可定制请求对象的请求头和请求数据。
- 修改在实际响应中提供了可定制响应对象的响应头和响应数据。

8.6.2　使用过滤器实现访问权限控制

(1) 将不同访问权限的页面置于不同的目录下。比如:将只允许管理员可见的内容都放在根目录的 admin 目录下。

(2) 编写过滤器类。

创建一个类,实现 Filter 接口,并且实现其中的 init()、doFilter()和 destroy()方法。doFilter()方法是 Filter 类的核心方法,希望过滤器完成的功能,都应该放到这个方法中。

doFilter()方法有 3 个参数:ServletRequest、ServletResponse 和 FilterChain,其中 ServletRequest 和 ServletResponse 为传递给方法的请求和响应参数,而 FilterChain 可以用来把请求和响应传递给下一个 Filter 或者其他 JSP/Servlet 等资源。

在 doFilter()中调用 FilterChain 的 doFilter()方法,它只有两个参数:ServletRequest 和 ServletResponse,通常只要将 Filter 的 doFilter()方法的前两个参数当作它的参数就可以了。

示例 8-21　实现管理员权限控制的过滤器 AdminFilter. java

```
package com.jyw.www.filter;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public void doFilter (ServletRequest request, ServletResponse response, FilterChain
chain)
        throws IOException, ServletException {
    request.setCharacterEncoding("gb18030");
    response.setCharacterEncoding("gb18030");
    HttpServletRequest httpreq= (HttpServletRequest)request;
```

```
    HttpServletResponse httpresp=(HttpServletResponse)response;
    Object obj=httpreq.getSession().getAttribute("authority");
    if(obj!=null)          从Session中取出用户角色，如果没有，则说明使用者没有登录
    {
    Integer authority=(Integer)obj;

    if(authority!=0){      管理员的权限值是0，如果不是0，则说明当前用户不是管理员
      httpreq.getRequestDispatcher("/privError.jsp").forward(httpreq,httpresp);
                                     跳转到权限错误提示页
    }else{
      chain.doFilter(httpreq, httpresp);

    }              如果是管理员，则转向下一个过滤器链上的下一个对象，
    }else          如果没有下一个对象，则通过过滤，跳转到请求访问的页面
    httpreq.getRequestDispatcher("/privError.jsp").forward(httpreq,httpresp);
}
public void init(FilterConfig arg0) throws ServletException {
        // TODO Auto-generated method stub
}
public void destroy() {
        // TODO Auto-generated method stub
}
```

读者可以参考示例 8-20，完成总店店员过滤器 HeadFilter.java 和分店店员过滤器 BranchFilter 的编写。

8.6.3 在 Web.xml 中配置过滤器

同 Servlet 的用法相似，要使用 Filter，也需要在 Web.xml 中进行配置。

示例 8-22 Web.xml 中 Filter 的配置

```
<filter>
    <filter-name>AdminFilter</filter-name>
    <filter-class>com.jyw.www.filter.AdminFilter</filter-class>
</filter>
<filter-mapping>
        <filter-name>AdminFilter</filter-name>
        <url-pattern>/admin/*</url-pattern>
</filter-mapping>           设置对/admin目录下的所有文件进行“过滤”，
                            该目录下的文件只能由管理员访问
<filter>
        <filter-name>HeadFilter</filter-name>
        <filter-class>com.jyw.www.filter.HeadFilter</filter-class>
</filter>
<filter-mapping>
        <filter-name>HeadFilter</filter-name>
        <url-pattern>/employee/*</url-pattern>
</filter-mapping>
                          设置对/employee目录下的所有文件进行“过滤”，
                          该目录下的文件只能由总店店员访问
```

```
<filter>
        <filter-name>BranchFilter</filter-name>
        <filter-class>com.jyw.www.filter.BranchFilter</filter-class>
</filter>
<filter-mapping>
        <filter-name>BranchFilter</filter-name>
        <url-pattern>/employee/branch/*</url-pattern>
</filter-mapping>
```

设置对/employee/branch目录下的所有文件进行“过滤”，该目录下的文件只能由分店店员访问，因为该目录为/employee的子目录，所以总店店员可访问该目录下的文件

现在，只需要将 JSP 文件分别放到相应的目录下，就可实现不同角色的权限控制了。例如，将实现将员工录入功能的 addEmployeeForm.jsp 文件移至根目录下的 admin 子目录中，只要不是管理员登录，对其进行访问时都会跳转到权限错误提示页，如图 8-12 所示。

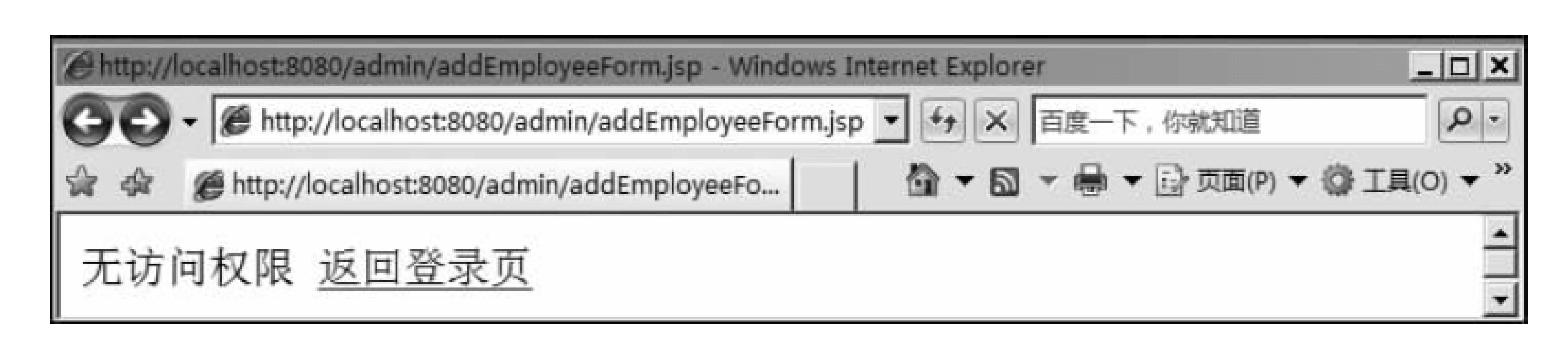


图 8-12 访问权限错误时的提示信息

8.7 知识扩展

8.7.1 Servlet 生命周期

Servlet 的生命周期是指一个 Servlet 从创建到结束的一系列步骤，分别由 Servlet 中相应的方法来代表。

1. init 方法

在服务器上最多只会驻留一个 Servlet 的实例，在第一次调用这个 Servlet 的时候，将会创建这个 Servlet 的实例。在创建这个 Servlet 实例的时候，HttpServlet 中的 init()方法会被调用，如果需要对这个 Servlet 做一些初始化，可以将初始化代码放在这个方法中。注意，这个方法只会被调用一次，不会对于每次的连接都调用它。

2. Service 方法

前面已经介绍了 doGet()和 doPost()方法，对于不同方式的请求，会自动去调用对应的方法。其实，如果客户端有一个对 Servlet 的请求发送过来，那么，服务器会产生一个新的线程，并且让它调用 Servlet 的 Service()方法。Service()方法会根据收到的客户端的

请求类型，决定调用 doGet()还是 doPost()甚至其他的 doXXX()方法。

3. doGet /doPost 方法

通过覆盖 HttpServlet 类中的 doGet()方法，可以处理浏览器端发送过来的 GET 请求。而 doPost()方法用于处理 POST 请求。

这两个方法的参数一样：HttpServletRequest 和 HttpServletResponse，它们也是分别代表客户端的请求和对客户端的响应。如果不能确定客户端的请求方式到底是 GET 方式还是 POST 方式，那么需要在 Servlet 中同时定义这两个方法。如果这两个方法的处理过程一样，那么只需要定义其中的一个，而让另外一个调用这个方法。

4. destroy 方法

如果因为某种原因，需要删除某个 Servlet 实例，那么，在删除这个 Servlet 实例之前，服务器会首先调用 destroy()方法。可以在这个方法中执行一些清理的动作，比如释放数据库连接、关闭打开的文件等。

8.7.2 Filter 生命周期

Filter 生命周期是指一个 Filter 从创建到结束的一系列步骤，分别由 Filter 中相应的方法来代表。

1. init 方法

初始化过滤器。

2. doFilter 方法

执行实际的过滤工作。在 doFilter()方法中，每个过滤器都接受当前的请求和响应，而 FilterChain 包含的过滤器则仍然要被处理。过滤器调用 chain.doFilter()将控制权传送给下一个过滤器。

3. destroy 方法

服务器调用 destory()方法结束过滤器。

8.7.3 Filter 链

Java 中的 Filter 并不是一个标准的 Servlet，它不能处理用户请求，也不能对客户端生成响应，主要用于对 HttpServletRequest 进行预处理，也可以对 HttpServletResponse 进行后处理，它是个典型的处理链。

Filter 有如下几个用处。

- 在 HttpServletRequest 到达 Servlet 之前，拦截客户的 HttpServletRequest。

- 根据需要检查 HttpServletRequest，也可以修改 HttpServletRequest 头和数据。
- 在 HttpServletResponse 到达客户端之前，拦截 HttpServletResponse。
- 根据需要检查 HttpServletResponse，可以修改 HttpServletResponse 头和数据。

Filter 有如下几个种类。

- 用户授权的 Filter：Filter 负责检查用户请求，根据请求过滤用户非法请求。
- 日志 Filter：详细记录某些特殊的用户请求。
- 负责解码的 Filter：包括对非标准编码的请求解码。
- 能改变 XML 内容的 XSLTFilter 等。

对同样的过滤目标，哪个过滤器在 Web.xml 中先配置，在过滤器链中就会先得到调用，在 Web.xml 中还可以配置 Filter 的一些初始化参数。下面举例说明 Filter 链的用法。

开发 Java Web 项目时，可能会出现中文乱码问题，比如本章前面描述的用户名和密码的显示结果可能会出现乱码，这是由于中文简体编码集不是浏览器默认的编码集的原因，这就需要程序明确地"告诉"用户的浏览器应该用什么编码方式来解析接收到的输出流。

在 Servlet 中，可以通过修改 Content-Type 完成，也就是在获得输出流之前，利用 setContentType()方法去设置 Content-Type 报头。

注意：*setContentType()方法必须在 HttpServletResponse 的 getWriter()方法获得输出流句柄之前调用。*

通过下面这条语句，就可以将输出的编码方式设置成 GB18030 编码方式。

```
response.setContentType("text/html;charset=GB18030");
```

但是，如果向客户端输出的中文是从表单中取出来的，还是会出现乱码的问题，这个问题也是因为编码方式不统一引起的。Servlet 在读取表单数据的时候，默认采用 iso-8859-1 的编码方式，此时，需要将 Servlet 读取表单数据的编码方式也设置为 GB18030，这个任务可以由 HttpServletRequest 的 setCharacterEncoding()方法来完成。在读取表单数据之前，调用这个方法来将编码设置成 GB18030 即可，示例如下：

```
request.setCharacterEncoding("GB18030");
```

下面编写编码格式过滤器来实现输入/输出编码格式的设置。

示例 8-23　编码格式过滤器 EncodingFilter.java

```
package com.jyw.www.filter;

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
```

```
import javax.servlet.ServletResponse;

public class EncodingFilter implements Filter {
    private FilterConfig filterConfig=null;
    private String encoding=null;
    public void destroy() {
        encoding=null;
    }

    public void doFilter(ServletRequest request, ServletResponse response,
    FilterChain chain) throws IOException, ServletException {
        String encoding=getEncoding();
        if (encoding==null) {
            encoding="GB18030";
        }
        request.setCharacterEncoding(encoding);
        response.setContentType("text/html;charset=GB18030");
        chain.doFilter(request, response);
    }

    public void init(FilterConfig filterConfig) throws ServletException {
        this.filterConfig=filterConfig;
        this.encoding=filterConfig.getInitParameter("encoding");
    }

    private String getEncoding() {
        return this.encoding;
    }

}
```

设置输入/输出编码格式为中文简体

读取Web.xml中的初始化参数

将下面的 EncodingFilter 配置写在其他 Filter 之间，就可以实现链式调用。读者可以使用 MyEclipse 的 Debug 功能来跟踪验证。

示例 8-24　Web. xml 中 EncodingFilter 的配置

```
<filter>
        <filter-name>encodingFilter</filter-name>
        <filter-class>com.jyw.www.filter.EncodingFilter</filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>GB18030</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>encodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

编码集初始化参数为GB18030

8.7.4　Cookie 属性的读写

在 JSP 中可以通过 cookie.setXXX 方法设置各种属性，用 cookie.getXXX 方法读出 Cookie 的属性，Cookie 的主要属性及其方法见表 8-1。

表 8-1　Cookie 读写各种属性的方法

类型	方　法　名	方 法 解 释
String	getComment()	返回 Cookie 中注释，如果没有注释将返回空值
String	getDomain()	返回 Cookie 适用的域名。使用 getDomain()方法可以指示浏览器把 Cookie 返回给同一域内的其他服务器，而通常 Cookie 只返回给发送它的服务器名字完全相同的服务器。注意域名必须以点开始(例如.yesky.com)
int	getMaxAge()	返回 Cookie 过期之前的最大时间，以秒计算
String	getName()	返回 Cookie 的名字
String	getPath()	返回 Cookie 适用的路径。如果不指定路径，Cookie 将返回给当前页面所在目录及其子目录下的所有页面
boolean	getSecure()	如果浏览器通过安全协议发送 Cookies，将返回 true 值；如果浏览器使用标准协议，则返回 false 值
String	getValue()	返回 Cookie 的值
int	getVersion()	返回 Cookie 所遵从的协议版本
void	setComment(String purpose)	设置 Cookie 中注释
void	setDomain(String pattern)	设置 Cookie 适用的域名
void	setMaxAge(int expiry)	以秒计算，设置 Cookie 过期的时间
void	setPath(String uri)	指定 Cookie 适用的路径
void	setSecure(boolean flag)	指出浏览器使用的安全协议，例如 HTTPS 或 SSL
void	setValue(String newValue)	Cookie 创建后设置一个新的值
void	setVersion(int v)	设置 Cookie 所遵从的协议版本

8.7.5　Cookie 的生存周期

Cookie 可以保持登录信息到用户下次与服务器的会话，换句话说，下次访问同一网站时，用户会发现不必输入用户名和密码就已经登录了(当然，不排除用户手工删除 Cookie)。而还有一些 Cookie 在用户退出会话的时候就被删除了，这样可以有效地保护个人隐私。

Cookie 在生成时就会被指定一个 Expire 值，这就是 Cookie 的生存周期，在这个周期内 Cookie 有效，超出周期 Cookie 就会被清除。有些页面将 Cookie 的生存周期设置为 0 或负值，这样在关闭页面时，就马上清除 Cookie，不会记录用户信息，使网页更加安全。

在 JSP 中，使用 setMaxAge(int expiry)方法来设置 Cookie 的生存周期。参数 expiry 应是一个整数，正值表示 Cookie 将在多少秒以后失效。注意这个值是 Cookie 将要存在

的最大时间，而不是 Cookie 现在的存在时间。负值表示当浏览器关闭时，Cookie 将会被删除。零值则是要删除该 Cookie。

8.7.6 MVC 设计模式

在本章之前，项目的功能模块都是采用 JSP＋JavaBean 的模式来完成的，通过将复杂的程序代码封装到 JavaBean 中，减少了 JSP 代码和网页标签混合使用的情况，在阅读和维护的时候会带来很大的方便；同时将公用的代码封装在 JavaBean 中，也提高了代码的可复用性。但是由于 JSP 页面中还是嵌入了较多的 Java 代码，当需要处理的业务逻辑非常复杂时，大量的 Java 代码使得 JSP 页面变得非常臃肿，前端的页面设计人员稍有不慎，就有可能破坏有关商业逻辑的代码。

本章引入了 Servlet 技术来完成了用户登录等模块的功能，采用了 MVC，即 Model-View-Controller 模式。在这种模式中，Servlet 充当了控制器（Controller 即 C）的角色，负责响应客户对业务逻辑的请求并根据用户的请求行为决定将要调用的 JSP 页面。

JSP 页面处于表现层，也就是视图（View 即 V）的角色。

JavaBean 负责数据的处理，也就是模型（Model 即 M）的角色。

在开发中，采用这种模式，可以具有更清晰的逻辑划分，能够有效地区分不同的角色，避免彼此间的互相影响，充分发挥每位开发人员的特长，较适合用于开发中到大型项目。

课 后 练 习

1. 将“佳衣屋”项目实现角色权限控制功能补充完整。

提示：参考示例 8-20，完成总店店员过滤器 HeadFilter. java 和分店店员过滤器 BranchFilter 的编写；按照示例 8-21 的 Web. xml 中的配置，合理组织 JSP 文件的目录结构，将各角色的受控使用 JSP 文件置于正确的目录中。

2. 将第 5～7 章中的各个任务用 JSP＋JavaBean＋Servlet 的模式来改写。

3. 本章主要学习内容为使用 Servlet 和 Filter 技术进行权限相关的安全控制。安全性设计和实现对于 Web 应用的正常运行至关重要！请读者结合第 1 章课后练习的要求完成“自选项目需求分析”中账号安全及权限控制等功能。

第 9 章　其他安全性设计

本章学习要点：

- 熟悉 Web 项目开发常见的安全性设计。
- 熟练掌握使用 JavaScript 进行输入的合法性检查。
- 熟练掌握正则表达式的用法。
- 熟练掌握 Java 监听器编写和部署的方法。

第 8 章对 Web 项目中与账号安全性控制相关的技术做了综合讲述，但 Web 项目的安全性设计不仅限于对账号的控制，本章将探讨其他需要考虑到的安全性设计及其实现技术。

9.1　任务一：对录入的员工信息做合法性检查

Web 应用中有许多信息需要用户从网上提交，用户可能由于疏忽，输入了一些格式不正确的信息，或者漏填了一些必不可少的信息，比如在员工信息的录入时，年龄中输入了非整数的字符串，或者漏填了账号和密码，如果不加检查就直接将用户输入的信息存入数据库中，会造成保存的信息失去使用价值，更严重的是还会给 Web 应用带来安全隐患，所以对于用户的输入信息一定要进行合法性检查。

任务目标：

在录入商品信息时需要对用户输入的信息进行合法性的检查。要求账号、密码、真实姓名必填，密码必须是 6 位的字母数字串，用户权限和所属分店必选。

技能训练：

- 使用 JavaScript 进行用户输入的合法性检查。
- 正则表达式的使用。

在录入商品信息时需要对用户输入的信息进行合法性的检查，可以使用 JavaScript 技术来完成，如示例 9-1 所示。

示例 9-1　带用户输入合法性检查的员工信息录入页 addEmployeeForm.jsp

```
<%@page language="java" import="java.util.* "
```

```
    contentType="text/html;charset=gb18030"%>

<center>
    员工信息录入
    <form name="form1" method="post" action="addEmployee.jsp"
        onsubmit="javascript: return addCheck()">
        用户名
        <input type="text" name="account">
        <br>
        密 码
        <input type="password" name="password">
        <br>
        姓 名
        <input type="text" name="name">
        <br>
        所属分店
        <select name="storenum">
            <option value="x">
                请选择所属分店
            </option>
            <option value="0">
                总店
            </option>
            <option value="1">
                一分店
            </option>
            <option value="2">
                二分店
            </option>
        </select>
        <br>
        授 权
        <select name="authority">
            <option value="x">
                请选择用户权限
            </option>
            <option value="0">
                管理员
            </option>
            <option value="1">
                总店店员
            </option>
            <option value="2">
                分店店员
            </option>
        </select>
        <br>
        <input type="submit" name="submit" value="添加" />
        <input type="reset" name="reset" value="重置" />
```

```
    </form>
</center>

<script>
function addCheck() {
    var account=document.form1.account.value;
    var password=document.form1.password.value;
    var name=document.form1.name.value;
    var storenum=document.form1.storenum.value;
    var storenum=document.form1.authority.value;
```

获取表单元素的值

```
    if (account=="") {
        alert("请输入账号!");
```

当检查到用户没有填写账号时，弹出警告框，提醒用户账号为必填项

```
        document.form1.account.focus();
```

将光标移至账号元素内闪烁，帮助用户快速定位出现问题的地方

```
        return false;
```

返回false，表单数据并不会被提交

```
    }

    var Regx=/^[A-Za-z0-9]{6}$/;
```

定义一个代表6位字母、数字串的正则表达式

```
    if (!Regx.test(password)) {
```

用这个正则表达式检查用户输入的密码

```
        alert("请输入 6 位的字母数字串密码!");
        document.form1.password.focus();
        return false;
```

如果密码不是6位的字母及数字串，则不提交表单数据，并要求用户重新填写密码

```
    }
    if (name=="") {
        alert("请输入真实姓名!");
        document.form1.name.focus();
        return false;
    }
    if (!(/^[0-9]{1}$/.test(storenum))) {
        alert("请选择所属分店!");
        document.form1.storenum.focus();
        return false;
    }
    if (!(/^[0-9]{1}$/.test(authority))) {
        alert("请选择用户权限!");
        document.form1.authority.focus();
        return false;
    }

}
</script>
```

9.2 任务二：过滤用户的恶意输入

Web 应用中，时常会遇到允许用户输入大段的文字，例如客户留言、对商品的评价等，如果某些网络用户借此机会恶意输入一段 HTML 标签或者 JS、JSP 代码，被不加处理地直接存储在数据库中，那么在这些信息被读取及显示的时候，会被当作有效代码来执行，而不是仅仅作为内容来输出，例如以下代码被当作用户留言存储在数据库中，当打开留言信息页时，会弹出一个显示“哈哈!”的警告框。

```
<script>
    alert("哈哈!");
<script>
```

任务目标：

将用户输入信息中的敏感的 HTML 字符转换为 HTML 转义符。

技能训练：

- Java 字符串的处理。
- Web 应用安全性设计意识的提高。

HTML 标签、JavaScript 程序块和 JSP 语法的构成都少不了“<”和“>”这两个符号，如果把此类敏感的字符转换成“<”和“>”这样的转义符，代码语法就会被破坏，不会再被执行了。下面编写一个字符转换器类，来完成本章任务二的要求。

示例 9-2 完成 HTML 转义符转换的 ConvertorUtil.java

```
package com.jyw.www.util;

import java.text.*;
import java.util.Date;
import java.io.*;

public class ConvertorUtil
{


    public static String convertToHtml(String s)
    {
        s=replace(s, "&", "&amp;");
        s=replace (s, "<", "&lt;");
        s=replace (s, ">", "&gt;");
        s=replace (s, "\t", "    ");
        s=replace (s, "\r\n", "\n");
        s=replace (s, "\n", "<br>");
        s=replace (s, "  ", "  ");
        s=replace (s, "'", "&#39;");
        s=replace (s, "\\", "&#92;");
```

将一些敏感字符替换成HTML转义符

```
        return s;
    }

    public static String replace(String s, String s1, String s2)
    {
        if(s==null)
        {
            return null;
        }
        StringBuffer stringbuffer=new StringBuffer();
        int i=s.length();
        int j=s1.length();
        int k;
        int l;
        for(k=0; (l=s.indexOf(s1, k))>=0; k=l+j)
        {
            stringbuffer.append(s.substring(k, l));
            stringbuffer.append(s2);
        }

        if(k<i)
        {
            stringbuffer.append(s.substring(k));
        }
        return stringbuffer.toString();
    }
}
```

可能包含敏感字符的数据在存入数据库之前，调用示例 9-2 中的 convertToHtml() 方法可过滤恶意的输入。

9.3　任务三：配置首页和全局错误提示页面

读者应该有这样的开发体验，即当发生页面找不到或者语法错时，会出现 JSP 错误提示信息，给出错误发生的位置和原因，其目的是帮助开发人员进行错误定位和调试。但是对于一个上线运行的 Web 应用仍然显示 JSP 的错误提示信息，会因为暴露系统的数据库、服务器、文件路径等信息给应用带来安全隐患。这时，需要为应用配置一个安全的错误提示页。

任务目标：

Web 应用系统发生 404 错误时，跳转到提示“网页正在建设中...”的提示页；Web 应用系统发生 500 错误时，跳转到提示“出错了，请通知系统管理员”的提示页。

技能训练：

Web. xml 中全局错误页的配置。

在 Web.xml 中配置全局的错误提示页面，代码如下。

示例 9-3 Web.xml 中全局错误提示页的配置

```
<error-page>
    <error-code>404</error-code>
    <location>/building.jsp</location>
</error-page>
<error-page>
    <error-code>500</error-code>
    <location>/error.jsp</location>
</error-page>
```

错误提示页的编写要设置 page 指令的 isErrorPage 属性为 true，表示当前页可以作为其他 JSP 页面的错误页面，并设置 response 的状态为“请求成功”，并中止错误状态的延续，示例如下：

```
<%@page language="java" contentType="text/html; charset=GBK"
            isErrorPage="true" pageEncoding="GBK"%>
<%response.setStatus(HttpServletResponse.SC_OK);%>
```

错误提示页的内容部分请读者自行完成。

9.4 任务四：统计系统在线人数

许多 Web 应用中会有统计当前在线人数的功能，使得用户或者管理员对该 Web 应用的使用情况有一个直观的了解，这个功能可以通过 HttpSessionListener 监听器来实现。Session 监听会话的创建或者销毁，它实现 HttpSessionListener 接口，并且实现接口的两个方法。

- void sessionCreated(HttpSessionEvent hse)：当一个 HttpSession 对象被创建时，将会调用这个方法；
- void sessionDestroyed(HttpSessionEvent hse)：当一个 HttpSession 超时或者调用 HttpSession 的 invalidate()方法让它销毁时，将会调用这个方法。

示例 9-4 实现了一个简单的在线人数统计。

示例 9-4 ActiveUserListener.java 实现在线人数统计

```
package com.jyw.www.listener;

import javax.servlet.*;
import javax.servlet.http.*;

public class ActiveUserListener implements HttpSessionListener {
    private static int sessionCount=0;

    public void sessionCreated(HttpSessionEvent se) {
```

```
        sessionCount++;
        }

    public void sessionDestroyed(HttpSessionEvent se) {
        if (sessionCount>0)
            sessionCount--;
    }
    public static int getActiveUsersCount()
    {
        return sessionCount;
    }

}
```

同 Servlet 和 Filter 一样，Listener 也需要在 Web. xml 中进行部署，如示例 9-5 所示。

示例 9-5　Web. xml 中监听器的部署

```
<listener>
  <listener-class>com.mysite.www.listener.Listener
    </listener-class>
</listener>
```

9.5　知识扩展

下面介绍一下正则表达式。

一个正则表达式(Regular Expression)就是由普通字符(例如字符 a～z)以及特殊字符(称为元字符)组成的文字模式。该模式描述了在查找文字主体时待匹配的一个或多个字符串。正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配。构建正则表达式可以通过查表来完成，表 9-1 是取自百度百科词条"正则表达式"中常用的元字符及其在正则表达式上下文中的行为，更多的元字符含义读者请自行查找资料。

表 9-1　常用的元字符及其在正则表达式上下文中的行为

字符	描　　述
\	将下一个字符标记为一个特殊字符，或一个原义字符，或一个后向引用，或一个八进制转义符。例如，'n'匹配字符"n"。'\n'匹配一个换行符。序列'\\'匹配"\"，而"\("匹配"("
*	匹配前面的子表达式零次或多次。例如，zo * 能匹配"z"以及"zoo"。 * 等价于{0,}
+	匹配前面的子表达式一次或多次。例如，'zo+'能匹配"zo"以及"zoo"，但不能匹配"z"。+ 等价于{1,}
?	匹配前面的子表达式零次或一次。例如，"do(es)?"可以匹配"do"或"does"中的"do"。? 等价于{0,1}
{*n*}	n 是一个非负整数。匹配确定的 n 次。例如，'o{2}'不能匹配"Bob"中的'o'，但是能匹配"food"中的两个 o

续表

字符	描　　述
{*n*,}	n 是一个非负整数。至少匹配 n 次。例如,'o{2,}'不能匹配"Bob"中的'o',但能匹配"foooood"中的所有 o。'o{1,}'等价于'o+'。'o{0,}'则等价于'o *'
{*n*,*m*}	m 和 n 均为非负整数,其中 n<=m。最少匹配 n 次且最多匹配 m 次。例如:"o{1,3}"将匹配"fooooood"中的前三个 o。'o{0,1}'等价于'o?'。请注意在逗号和两个数之间不能有空格
x|*y*	匹配 x 或 y。例如,'z|food'能匹配"z"或"food"。'(z|f)ood'则匹配"zood"或"food"
[*xyz*]	字符集合。匹配所包含的任意一个字符。例如,'[abc]'可以匹配"plain"中的'a'
[^*xyz*]	负值字符集合。匹配未包含的任意字符。例如,'[^abc]'可以匹配"plain"中的'p'
[*a*-*z*]	字符范围。匹配指定范围内的任意字符。例如,'[a-z]'可以匹配'a'到'z'范围内的任意的小写字母字符
[^*a*-*z*]	负值字符范围。匹配任何不在指定范围内的任意字符。例如,'[^a-z]'可以匹配任何不在'a'到'z'范围内的任意字符
\b	匹配一个单词边界,也就是指单词和空格间的位置。例如,'er\b'可以匹配"never"中的'er',但不能匹配"verb"中的'er'
\B	匹配非单词边界。'er\B'能匹配"verb"中的'er',但不能匹配"never"中的'er'
\c*x*	匹配由 x 指明的控制字符。例如,\cM 匹配一个 Control-M 或回车符。x 的值必须为 A~Z 或 a~z 之一。否则,将 c 视为一个原义的'c'字符
\d	匹配一个数字字符,等价于[0-9]
\D	匹配一个非数字字符,等价于[^0-9]
\f	匹配一个换页符,等价于\x0c 和\cL
\n	匹配一个换行符,等价于\x0a 和\cJ
\r	匹配一个回车符,等价于\x0d 和\cM
\s	匹配任何空白字符,包括空格、制表符、换页符等。等价于[\f\n\r\t\v]
\S	匹配任何非空白字符,等价于[^\f\n\r\t\v]
\t	匹配一个制表符,等价于\x09 和\cI
\v	匹配一个垂直制表符,等价于\x0b 和\cK
\w	匹配包括下画线的任何单词字符,等价于'[A-Za-z0-9_]'
\W	匹配任何非单词字符,等价于'[^A-Za-z0-9_]'

课后练习

1. 为“佳衣屋”的任何来自用户的数据添加合法性安全控制。
2. 将本章讲授的安全性设计运用到“自选项目”中。



第四部分

数 据 分 析



第 10 章　图表的生成

本章学习要点：

- 熟练掌握将 Web 数据导出到 Excel 中的方法。
- 熟练掌握将 Web 数据导出到 Word 中的方法。
- 熟练掌握使用 JFreeChart 绘制柱图的方法。
- 熟练掌握使用 JFreeChart 绘制饼图的方法。
- 熟练掌握使用 JFreeChart 绘制折线图的方法。

经过前三部分的建设后，“佳衣屋”信息管理系统的日常管理功能基本实现，随着系统的上线运行，因入库、销售等行为，在数据库中“收集”了许多经营数据，通过对经营数据的分析，企业管理者可以及时了解经营现状，发现问题，辅助未来经营管理的决策。

10.1　任务一：将数据导出至 Excel 中

Microsoft Excel 是微软公司 Microsoft Office 办公软件的组件之一，它可以进行各种数据的处理、统计分析和辅助决策操作，广泛地应用于管理、统计财经、金融等众多领域，可以将数据表中的信息导出到 Excel 中，再使用 Excel 的内置功能完成数据分析。

任务目标：

将各分店的调货请求信息导出到 Excel 表中。

技能训练：

视图的创建。

可以通过设置 MIME 类型为来完成任务一的要求。MIME 类型就是设定某种扩展名的文件用一种应用程序来打开的方式类型，当该扩展名文件被访问的时候，浏览器会自动使用指定应用程序来打开。在 JSP 文件中通过设置 page 指令的 contentType 属性的值来指定 MIME 类型，用 Excel 打开的. xls 文件的 MIME 类型为“application/msexcel”如示例 10-1 所示。

示例 10-1 showExcel.jsp

```
<%@page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@page  import="java.util.*,com.jyw.www.dao.*,com.jyw.www.entity.*"%>
<%@page contentType="application/msexcel"%>
                                        设置标题字体
<%
  response.setHeader("Content-disposition","attachment; filename=showExcel.xls");
%>                                                      设置输出.xls文件的名字
<center>商品调货情况</center>
<table border=1 align="center" bgColor="#ffffff">
    <tr>
        <td align=center><B>序号</B></td>
        <td align=center><B>款号</B></td>
        <td align=center><B>颜色</B></td>
        <td align=center><B>类别</B></td>
        <td align=center><B>码数</B></td>
        <td align=center><B>数量</B></td>
        <td align=center><B>调货日期</B></td>
        <td align=center><B>处理日期</B></td>
        <td align=center><B>状态</B></td>
    </tr>
    <%
        OrderinfoDAO orderinfoDao=new OrderinfoDAO();
        List<Orderinfo>orderinfoList=orderinfoDao.findAll();
        for (Orderinfo orderinfo:orderinfoList) {
    %>
    <tr>
        <td align=center><%=orderinfo.getId()%></td>
        <td align=center><%=orderinfo.getGoodsid()%></td>
        <td align=center><%=orderinfo.getColor()%></td>
        <td align=center><%=orderinfo.getName()%></td>
        <td align=center><%=orderinfo.getSize()%></td>
        <td align=center><%=orderinfo.getQuantity()%></td>
        <td align=center><%=orderinfo.getApplydate()%></td>
        <td align=center>
        <%=orderinfo.getProcessdate()!=null?orderinfo.getProcessdate():""%>
        </td>
        <td align=center><%=orderinfo.getOrderstatus()%></td>
    <%
      }
    %>
    </tr>
</table>
```

运行 showExcel.jsp 时会弹出一个窗口，询问用户是“保存”还是直接“打开”此 xls 文件，任何一种方式下，都可以得到如图 10-1 所示的商品调货情况的 Excel 文件。



商品调货情况								
序号	款号	颜色	类别	码数	数量	调货日期	处理日期	状态
1	JYW12CKN1	水洗黑	牛仔裤	S	10	2013/2/22	2013/2/22	待发货
2	JYW12CKN1	水洗黑	牛仔裤	M	20	2013/2/22	2013/4/5	已发货
9	JYW12CKN1	水洗黑	牛仔裤	M	2	2013/3/31	2013/4/5	已取消
8	JYW12CKB0	蓝	牛仔裤	S	2	2013/3/31		待处理

图 10-1　从 Web 导出的 Excel 文件

10.2　任务二：将数据输出至 Word 中

微软公司 Office 系列办公组件中的另一个软件 Word，是目前世界上最流行的文字编辑软件，拥有强大的图片混排和表格制作功能。Web 数据也常常因为存档及再次编辑、打印等原因需要导出到 Word 中。

任务目标：

将各分店的调货请求信息导出到 Word 文件中。

技能训练：

视图的创建。

与任务一相似，将 MIME 类型设置为 application/msword，就可以将 Web 数据列表导出到 Word 中了，如图 10-2 所示。



序号	款号	颜色	类别	码数	数量	订购日期	处理日期	状态
1	JYW12CKN1	水洗黑	牛仔裤	S	10	2013-02-22	2013-02-22	待发货
2	JYW12CKN1	水洗黑	牛仔裤	M	20	2013-02-22	2013-04-05	已发货
9	JYW12CKN1	水洗黑	牛仔裤	M	2	2013-03-31	2013-04-05	已取消
8	JYW12CKB0	蓝	牛仔裤	S	2	2013-03-31		待处理

图 10-2　从 Web 导出的 Word 文件

示例 10-2　showWord.jsp 中指定 MIME 类型和导出文件名的语句

```
<%@page contentType="application/msword" %>
<%
```

```
    response.setHeader("Content-disposition","attachment;filename=showWord.doc");
%>
```

10.3 任务三：各分店销量的柱图统计

表格列出的只是未经处理和分析的原始数据，不像图形图像的数据表示形式那样可以直观地反映数据中蕴藏的变化和规律。JFreeChart 是一组功能强大、灵活易用的 Java 绘图第三方组件。在 Web 中得到广泛的应用，使用它可以生成多种报表，包括柱状图、饼图、折线图等。

柱形图是将一个或多个类别的数据进行可视化的好方法，用来显示一段时间内数据的变化或者描述各项目之间的数据比较，直观地表现出各类数据点的差别，它是最常用的图表类型之一。

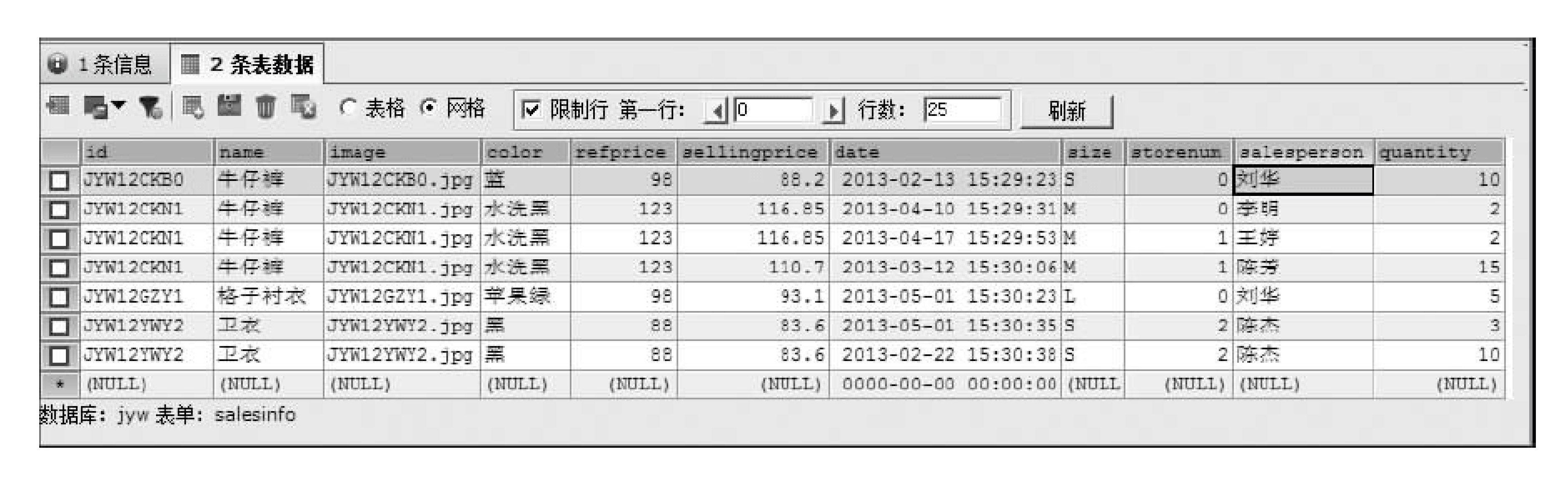
任务目标：

为佳衣屋项目添加的月销售量统计柱图。

技能训练：

- 视图的创建。
- 实体类的编写规范。
- DAO 类中 findAll()方法的编写规范。
- 使用 JFreeChart 绘制柱图。

第 3 章任务四中，创建了视图 salesinfo，如图 10-3 所示，这个视图中记录了所发生的每一笔交易，月销量可以根据此视图来进行统计。



id	name	image	color	refprice	sellingprice	date	size	storenum	salesperson	quantity
JYW12CKB0	牛仔裤	JYW12CKB0.jpg	蓝	98	88.2	2013-02-13 15:29:23	S	0	刘华	10
JYW12CKN1	牛仔裤	JYW12CKN1.jpg	水洗黑	123	116.85	2013-04-10 15:29:31	M	0	李明	2
JYW12CKN1	牛仔裤	JYW12CKN1.jpg	水洗黑	123	116.85	2013-04-17 15:29:53	M	1	王婷	2
JYW12CKN1	牛仔裤	JYW12CKN1.jpg	水洗黑	123	110.7	2013-03-12 15:30:06	M	1	陈芳	15
JYW12GZY1	格子衬衣	JYW12GZY1.jpg	苹果绿	98	93.1	2013-05-01 15:30:23	L	0	刘华	5
JYW12YWY2	卫衣	JYW12YWY2.jpg	黑	88	83.6	2013-05-01 15:30:35	S	2	陈杰	3
JYW12YWY2	卫衣	JYW12YWY2.jpg	黑	88	83.6	2013-02-22 15:30:38	S	2	陈杰	10
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	0000-00-00 00:00:00	(NULL	(NULL)	(NULL)	(NULL)

数据库: jyw 表单: salesinfo

图 10-3　视图 salesinfo 中的记录

再创建一个视图 salesstatistic，脚本如示例 10-3 所示，这个视图将按月统计销售量和销售额。

示例 10-3　视图 salesstatistic 的 SQL 脚本

```
DELIMITER $$

USE 'jyw'$$
```

```
DROP VIEW IF EXISTS 'salesstatistic'$$

CREATE ALGORITHM=UNDEFINED DEFINER='root'@'localhost' SQL SECURITY
                                        DEFINER VIEW 'salesstatistic' AS
SELECT
  DATE_FORMAT('salesinfo'.'date','%Y-%m') AS 'month',
  SUM('salesinfo'.'quantity') AS 'quantity',
  SUM(('salesinfo'.'sellingprice' * 'salesinfo'.'quantity')) AS 'sales'
FROM 'salesinfo'
GROUP BY DATE_FORMAT('salesinfo'.'date','%Y-%m')
ORDER BY DATE_FORMAT('salesinfo'.'date','%Y-%m')$$

DELIMITER;
```

salesstatistic 视图创建完成后，其中的记录如图 10-4 所示。

按照第 5 章中讲授的方法，为 salesstatistic 视图创建相应的实体 Bean SalesStatistic 和 DAO 类 SalesStatisticDAO，并为 DAO 类添加 findAll()方法，将视图中的相关数据读出后就可以利用 JFreeChart 绘制图表了。

使用 JFreeChart，需要在 Web. xml 文件中增加如下配置，参考示例 10-4。



month	quantity	sales
2013-02	20	1718
2013-03	15	1660.5
2013-04	4	467.4
2013-05	8	716.3
(NULL)	(NULL)	(NULL)

图 10-4　视图 salesstatistic 中的记录

示例 10-4　Web. xml 中 JFreeChart 的配置

```
<servlet>
    <servlet-name>DisplayChart</servlet-name>
    <servlet-class>org.jfree.chart.servlet.DisplayChart</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>DisplayChart</servlet-name>
    <url-pattern>/DisplayChart</url-pattern>
</servlet-mapping>
```

下面编写月销量统计的柱形图的显示文件 bar. jsp。

示例 10-5　bar. jsp

```
<%@page contentType="text/html;charset=GB18030"%>
<%@page
    import="org.jfree.chart.*,
    org.jfree.chart.plot.*,
    org.jfree.chart.servlet.ServletUtilities,
    org.jfree.chart.title.*,
    org.jfree.data.category.DefaultCategoryDataset,
    java.awt.Font,
    org.jfree.data.category.CategoryDataset,
    org.jfree.data.general.DatasetUtilities,
    org.jfree.chart.axis.*,
    javax.naming.InitialContext,
    javax.sql.DataSource,
```

引入JFreeChart所需的包

```
    java.sql.*,
    com.jyw.www.dao.*,
    java.util.*,
    com.jyw.www.entity.* "
%>
<%
    DefaultCategoryDataset dataset=new DefaultCategoryDataset();
    SalesStatisticDAO salesStatisticDAO=new SalesStatisticDAO();
    List<SalesStatistic>salesList=salesStatisticDAO.findAll();
    for(SalesStatistic salesstatistic: salesList) {
      dataset.addValue(salesstatistic.getQuantity(),"",salesstatistic.getMonth());

}                                        将月份和销售量信息添加到数据集中

    JFreeChart chart=ChartFactory.createBarChart3D("销量量统计柱图", "",
            "销量", dataset, PlotOrientation.VERTICAL, false, false,
            false);
                                         对数据集应用柱图主题样式
    TextTitle textTitle=chart.getTitle();
    textTitle.setFont(new Font("宋体", Font.BOLD, 20));    设置标题字体
    LegendTitle legend=chart.getLegend();
    if (legend !=null) {
        legend.setItemFont(new Font("宋体", Font.BOLD, 20));
    }                                                      设置图例的字体
    CategoryPlot plot=(CategoryPlot) chart.getPlot();
    CategoryAxis domainAxis=plot.getDomainAxis();


                                                   设置x轴上的字体
    domainAxis.setTickLabelFont(new Font("宋体", Font.BOLD, 20));
                                                   设置x轴上标题的字体
    domainAxis.setLabelFont(new Font("宋体", Font.BOLD, 20));

    ValueAxis valueAxis=plot.getRangeAxis();//柱形图的 y 轴   设置y轴上的字体
    valueAxis.setTickLabelFont(new Font("宋体", Font.BOLD, 20));
    valueAxis.setLabelFont(new Font("宋体", Font.BOLD, 20));
                                         设置y轴上标题的字体

    String filename=ServletUtilities.saveChartAsPNG(chart,500,300,null,session);
                                         将柱图统计存储为png文件
    String graphURL=request.getContextPath()
            +"/DisplayChart?filename="+filename;
%>                                       获取柱图的图片文件URL
<center>
    <img src="<%=graphURL%>" width=500 height=300 border=0
        usemap="#<%=filename %>">               显示柱图图片
</center>
```

bar.jsp 的执行效果如图 10-5 所示。

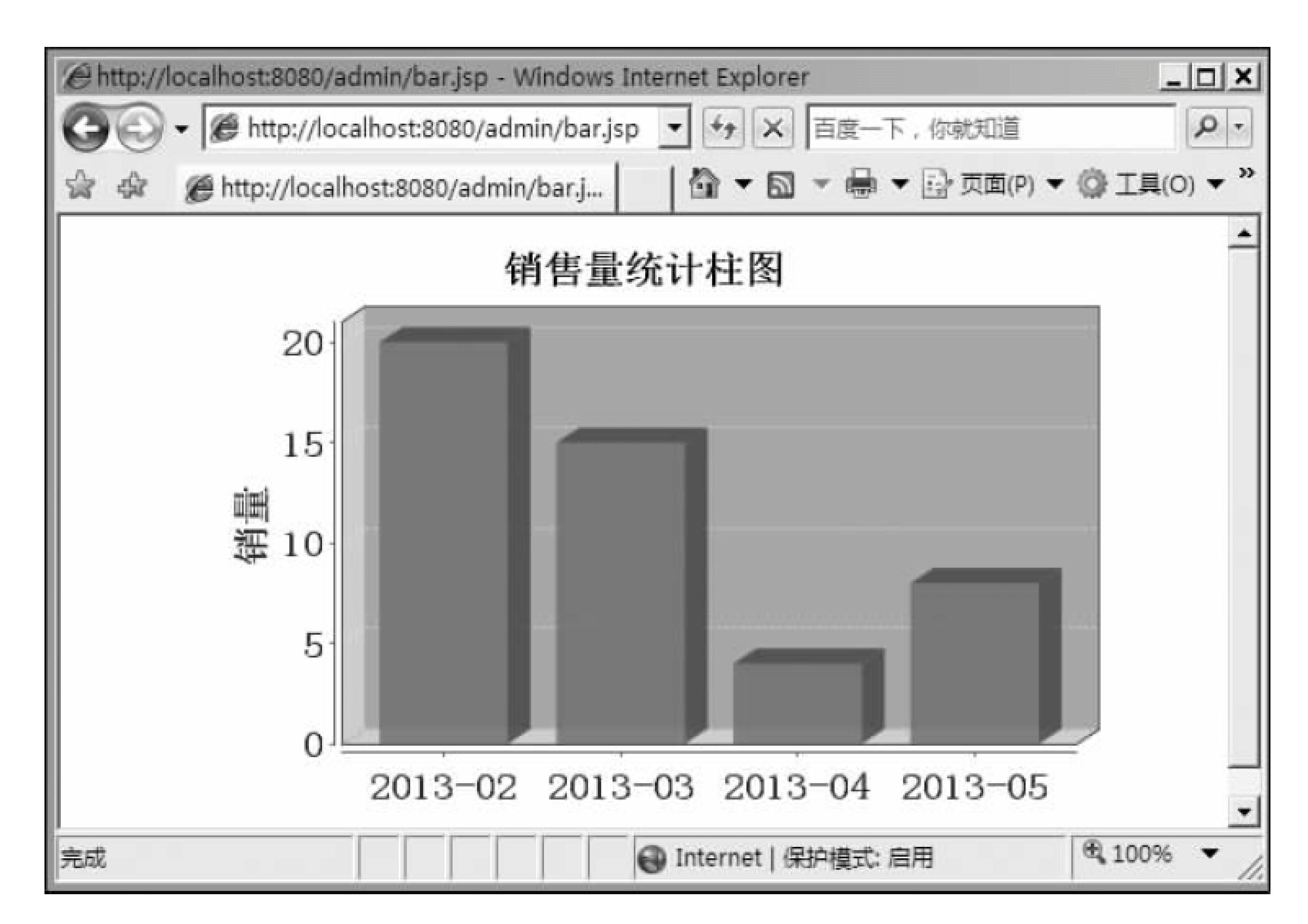


图 10-5　月销量统计柱图

10.4　任务四：商品销售额折线图

折线图是将一系列数据连接起来，可以表现数据的动态变化。

任务目标：

为佳衣屋项目添加月销售额统计折线图。

技能训练：

使用 JFreeChart 绘制折线图。

下面编写月销售额统计的折线图的显示文件 salesline.jsp.jsp。

示例 10-6　salesline.jsp

```
<%@page contentType="text/html;charset=GB18030"%>
<%@page
    import="org.jfree.chart.*,
    org.jfree.chart.plot.*,
    org.jfree.chart.servlet.ServletUtilities,
    org.jfree.chart.title.*,
    org.jfree.data.category.DefaultCategoryDataset,
    java.awt.Font,
    org.jfree.data.category.CategoryDataset,
    org.jfree.data.general.DatasetUtilities,
    org.jfree.chart.axis.*,
    javax.naming.InitialContext,
```

```
    javax.sql.DataSource,
    java.sql.*,
    com.jyw.www.dao.*,
    com.jyw.www.entity.*,
    com.jyw.www.util.*"%>
<%
    Test ld=new Test();
    List<String[]>list=new ArrayList<String[]>();
    SalesStatisticDAO salesStatisticDao=new SalesStatisticDAO();
    List<SalesStatistic>salesList=salesStatisticDao.findAll();
    DefaultCategoryDataset dataset=new DefaultCategoryDataset();
    for (SalesStatistic salesstatistic : salesList) {
    dataset.addValue(Double.valueOf(String.valueOf(salesstatistic.
            getSales()),"",salesstatistic.getMonth());
    }
    st.setRegularFont(new Font("宋体", Font.PLAIN, 15));


    JFreeChart chart=ChartFactory.createLineChart(      对数据集应用折线图主题样式
            "销售量统计折线图",                          设置标题
            "月份",                                      设置x轴标题

            "销售额",                                    设置y轴标题
            dataset, // data
            PlotOrientation.VERTICAL,                   显示方向
            false,
            true,                                       确定是否显示图例，false为不显示
            false
            );
    CategoryPlot line=chart.getCategoryPlot();
    TextTitle textTitle=chart.getTitle();
    textTitle.setFont(new Font("宋体", Font.BOLD, 20));
    LegendTitle legend=chart.getLegend();
    if (legend !=null) {
          legend.setItemFont(new Font("宋体", Font.BOLD, 20));
    }
    CategoryPlot plot=(CategoryPlot) chart.getPlot();
    CategoryAxis domainAxis=plot.getDomainAxis()
    domainAxis.setTickLabelFont(new Font("宋体", Font.BOLD, 20));
    domainAxis.setLabelFont(new Font("宋体", Font.BOLD, 20));
    ValueAxis valueAxis=plot.getRangeAxis();
    valueAxis.setTickLabelFont(new Font("宋体", Font.BOLD, 20));
    valueAxis.setLabelFont(new Font("宋体", Font.BOLD, 20));

    String filename="";
    try {
        filename=ServletUtilities.saveChartAsPNG(chart, 600, 300,
                null, request.getSession());
    } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        String graphURL=request.getContextPath()
                +"/DisplayChart?filename="+filename;
%>
<center>
    <img src='<%=graphURL%>' width=600 height=300 border=0
        usemap='#<%=filename%>'>
</center>
```

salesline.jsp 的执行效果如图 10-6 所示。

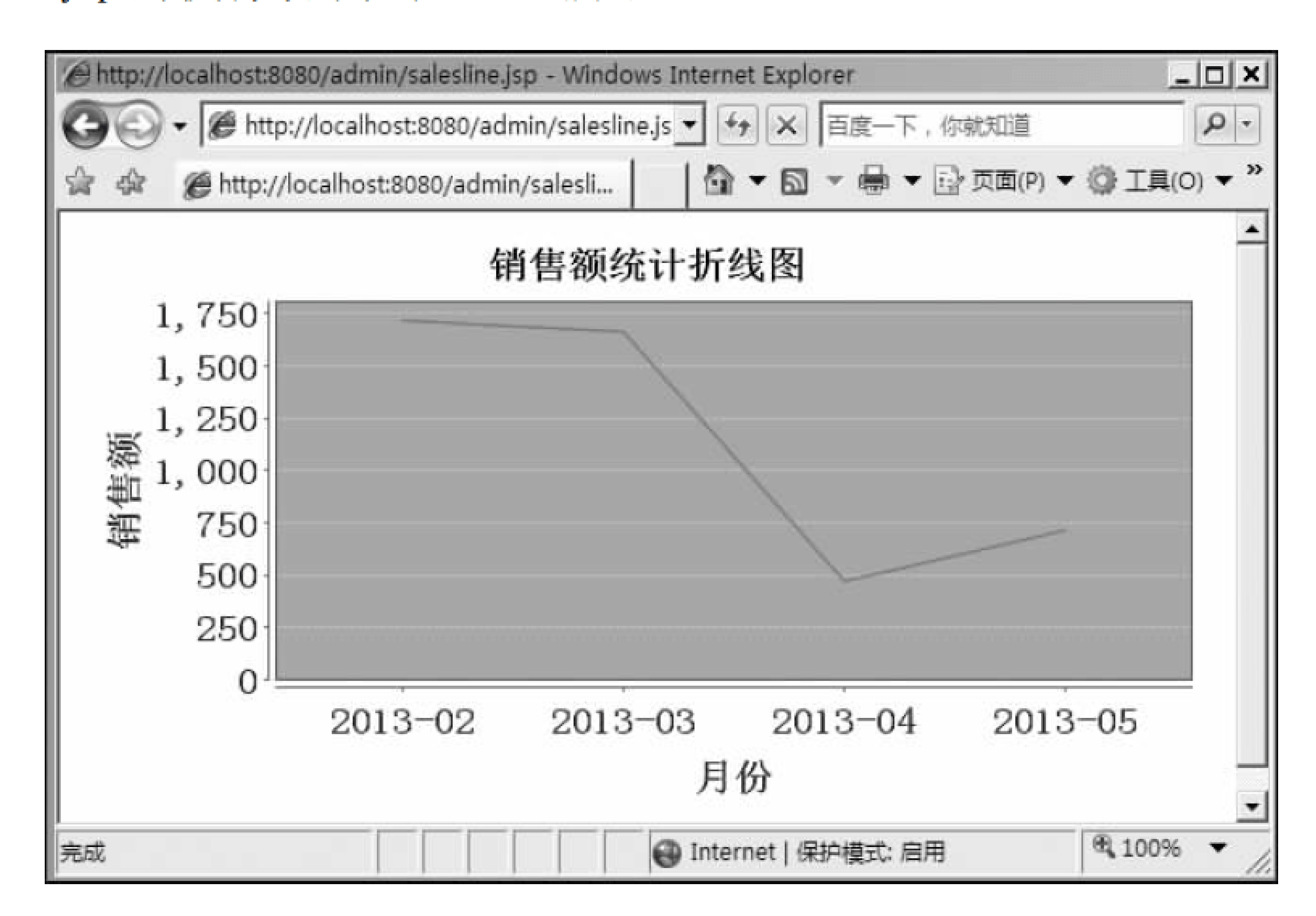


图 10-6　月销售额折线图

10.5　任务五：各类商品销量的饼图统计

饼图是一个划分为几个扇形的圆形统计图表，用于描述量或百分比之间的相对关系。

任务目标：

为佳衣屋项目添加销售量分布饼图。

技能训练：

- 视图的创建。
- 实体类的编写规范。
- DAO 类中 findAll()方法的编写规范。
- 使用 JFreeChart 绘制饼图。

创建一个视图 catigoriesStatistic，脚本如示例 10-7 所示，这个视图将按月统计销售

量和销售额。

示例 10-7 视图 catigoriesStatistic 的 SQL 脚本

```
DELIMITER $$

USE 'jyw'$$
DROP VIEW IF EXISTS 'catigoriesStatistic'$$
CREATE ALGORITHM=UNDEFINED DEFINER='root'@'localhost' SQL SECURITY DEFINER
VIEW 'catigoriesstatistic' AS
SELECT
  'salesinfo'.'name' AS 'id',
  SUM('salesinfo'.'quantity') AS 'quantity'
FROM 'salesinfo'
GROUP BY 'salesinfo'.'name'$$
DELIMITER;
```

catigoriesStatistic 视图创建完成后，其中的记录如图 10-7 所示。



图 10-7 视图 catigoriesStatistic 中的记录

按照第 5 章中讲授的方法，为 catigoriesStatistic 视图创建相应的实体 Bean CatigoriesStatistic 和 DAO 类 CatigoriesStatisticDAO，并为 DAO 类添加 findAll()方法，将视图中的相关数据读出后，就可以利用 JFreeChart 绘制图表了。

pie.jsp 文件中用于显示各类型销量统计的饼形图的部分代码如下。

示例 10-8 pie.jsp

```
<%@page contentType="text/html;charset=GBK"%>
<%@page
     import="org.jfree.chart.*,org.jfree.chart.plot.PiePlot,org.jfree.data.
            general.DefaultPieDataset,org.jfree.chart.servlet.ServletUtilities,
            org.jfree.chart.title.*,java.awt.Font,com.jyw.www.dao.*,com.jyw.
            www.entity.*,java.util.*"%>
<%
    DefaultPieDataset dataset=new DefaultPieDataset();
    CatigoriesStatisticDAO catigoriesStatisticDao=new CatigoriesStatisticDAO();
    List<CatigoriesStatistic>catigoriesStatisticList=
    catigoriesStatisticDao.findAll();
    For(CatigoriesStatistic catigoriesStatistic: catigoriesStatisticList){
        dataset.setValue(catigoriesStatistic.getId(),
        catigoriesStatistic.getQuantity());
    }
    JFreeChart chart=ChartFactory.createPieChart3D("各类型销量分布图",
            dataset, true, false, false);
                                                    对数据集应用饼图样式
    TextTitle textTitle=chart.getTitle();
    textTitle.setFont(new Font("宋体", Font.BOLD, 20));
```

```
    LegendTitle legend=chart.getLegend();
    if (legend !=null) {
        legend.setItemFont(new Font("宋体", Font.BOLD, 20));
    }
    PiePlot pieplot=(PiePlot) chart.getPlot();
    pieplot.setLabelFont(new Font("宋体", 0, 12));
    Font LegendFont=new Font("楷体", Font.PLAIN, 18);
    pieplot.setNoDataMessage("无数据显示");
    pieplot.setCircular(false);
    pieplot.setLabelGap(0.02D);
    String filename=ServletUtilities.saveChartAsPNG(chart, 500, 300,
            null, session);
    String graphURL=request.getContextPath()+"/DisplayChart?
    filename="+filename;
%>
<center>
<img src="<%=graphURL%>" width=500 height=300 border=0
    usemap="#<%=filename %>">
    </center>
```

salesline.jsp 的执行效果如图 10-8 所示。

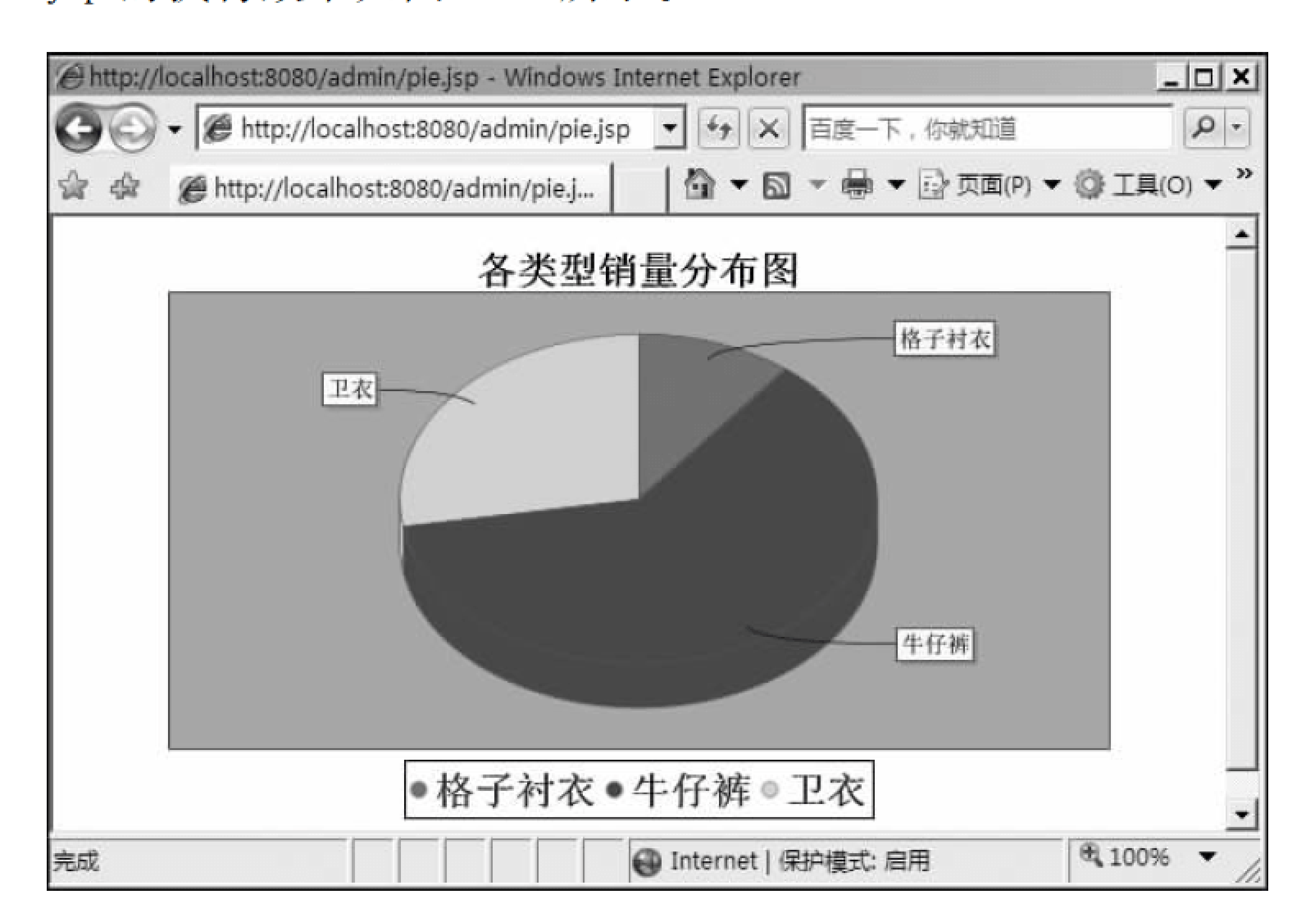


图 10-8　分类销售量的统计饼图

课 后 练 习

1. 将公司员工信息导出到 Excel 中。

要求：结合实际应用，添加员工工资信息表 salary，将员工的月工资导出到 Excel 中，形成工资表。

提示：参考本章任务一，完成本练习。

2. 为分店添加分店销售量折线图统计的功能。

要求：分店店员登录系统后，可查看本分店的周销售量和销售额的折线图。

提示：参考本章任务四，完成本练习。

3. 为管理员添加分店销售额柱图。

要求：管理员登录系统后，可查看指定月份的各分店销售额的柱图。

提示：参考本章任务三，完成本练习。

4. 本章主要学习内容为数据报表的生成。可按照第一章课后练习的要求完成“自选项目需求分析”项目中应有的数据报表分析。

参 考 文 献

1. 陈雪莲. JSP 程序设计教程 . 北京:清华大学出版社,北京交通大学出版社,2008
2. 布朗. JSP 编程指南. 白雁译. 北京:电子工业出版社,2004
3. 李兴华. Java Web 开发实战经典(基础篇). 北京:清华大学出版社,2010
4. 卢瀚等. Java Web 开发实战 1200 例(第 1 卷). 北京:清华大学出版社,2011
5. 李明革等. Java Web 应用教程:网上购物系统的实现. 北京:人民大学出版社,2011
6. 聂哲. JSP 动态 Web 技术实例教程. 北京:高等教育出版社,2009